Digit Classification using PCA based feature space reduction to Eigendigits

Harshal Priyadarshi
harshal@cs.utexas.edu

February 9, 2016

1 Overview

Individual pixels are the lowest level uncorrelated features of an image. However, the number of such features can quickly grow to be intractable for real image corpuses. For instance a HD image of size($720x1280$) has $921,600$ feature pixels. Running any Machine learning algorithm on such huge feature set for a single sample can be painfully slow, no matter how efficient is the algorithm. Thus, the task in hand is to find a relatively smaller set of features, that can represent the sample while retaining most of the information that the original image has. Principal Component Analysis (PCA)[1] is a novel algorithm that has proven to be beneficial in this case and has been theoretically proven to retain the maximum amount of information possible when reducing to a lower dimensional feature space.

2 Notation

- x = total number of pixels in an image (= 784 for 28 x 28 image)
- t = number of training samples used for finding eigendigits
- n = Number of training samples($= 60,000$)
- A = sample from training data of dimension$\mathbb{R}^{(x,t)}$
- Σ = covariance matrix of A = AA^*
- M = mean image of the t training images used
- V = eigenvector matrix $\in \mathbb{R}^{(x,t)}$ for covariance matrix Σ.
- k = number of nearest neighbours for K-Nearest Neighbors algorithm
- fit_factor = ratio of the training data for KNN[2] vs the training data for eigendigits

3 Concepts

The eigenvalues of the covariance matrix,Σ is basically, squared singular values of the data matrix, A. This can be easily shown. Say, the SVD[3] of A is $U\Sigma V^*$, then

$$AA^* = U\Sigma\Sigma^*U^* \tag{1}$$

However, the eigenvalue decomposition[4] of AA^* can be written as,

$$AA^* = X^{-1}\Lambda X \tag{2}$$

where, X are the eigenvectors and Λ are the eigenvalues. Comparing equation (1) and (2) we get our desired relation.
Now it has been proved that the top few singular values of A(and thus, top few eigenvalues of Σ) preserves most of the information in A. Thus taking top t eigenvectors and projecting the original feature vector to this new subspace, preserves most of the information in A, but at the same time lowers the feature space to, a more tractable $\mathbb{R}^{(t,n)}$, where $t \ll x$.
Now the problem is in finding the eigenvectors for the covariance matrix. However this if done with a straightforward approach in literature[4] for the covariance matrix, will soon become intractable due to huge computational complexity. We can see this, with the following argument.
Eigenvector representation of covariance matrix $\Sigma = AA^*$ is

$$\Sigma V = V\Lambda \tag{3}$$

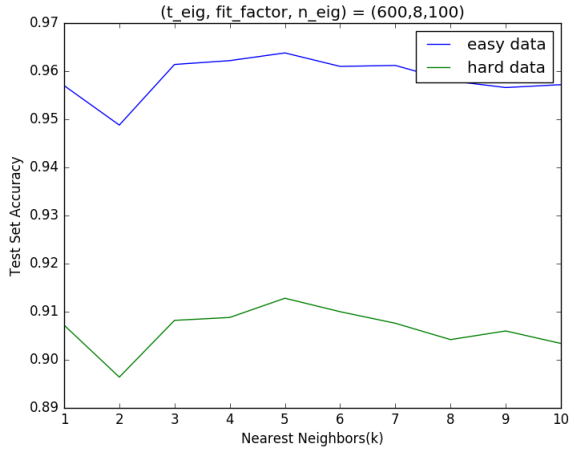
Now, Σ is an (x, x) matrix, which for a normal size image (say a 720 x 1280) will be a 921000 x 921000 matrix. Caluclating the eigenvectors for this matrix will be almost impossible and/or extremely slow. However, there is a trick. We can get away with calulating the eigenvectors for the smaller t x t matrix, A^*A, and then using its results to obtain the top t-eigenvectors for the original covariance matrix. This can be shown as follows. Assume that λ,v is eigenvalue, eigenvector pair for A^*A

$$A^*Av = \lambda v \tag{4}$$
$$AA^*Av = \lambda Av \tag{5}$$

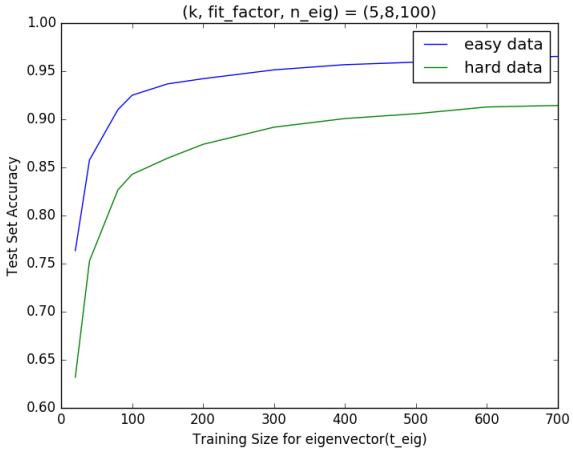
Thus, if λ, v are the eigenpairs for the reduced matrix, then λ, Av are the eigenpairs for the original huge covariance matrix. Thus, once we get the eigenpairs for the reduced t x t matrix, all that remains is to get the original eigenvectors fromm the reduced ones, by pre-multiplying the eigenvector of A^*A by A.
However, we only get t eigenvectors of the original set of x eigenvectors. But it can be proved mathematically that, these t eigenvectors are the top t eigenvectors with respect to the eigenvalues.

4 Algorithm

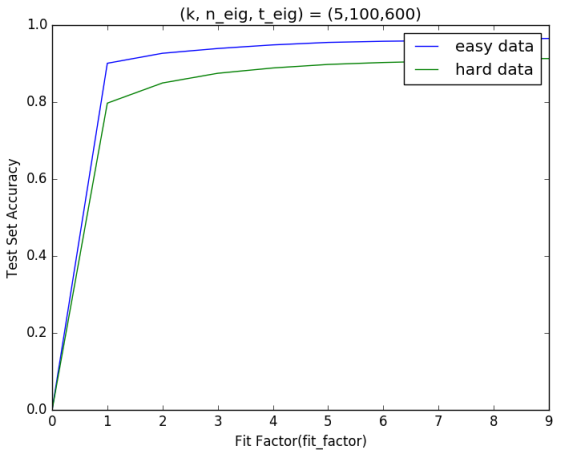
1. Randomly choose set of t training samples out of n samples to create the matrix A, for constructing eigenspace.



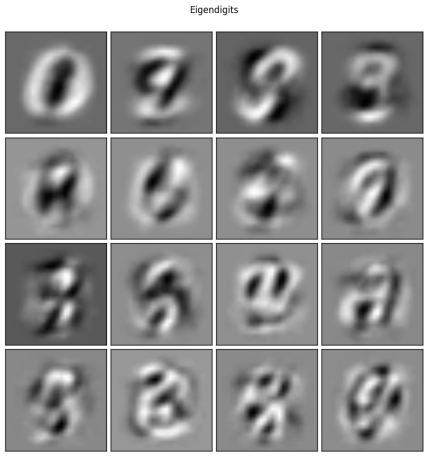
(a) Accuracy vs Nearest Neighbors



(b) Accuracy vs Eigenspace dimension



(c) Accuracy vs fit factor



(d) 1^{st} 16 eigendigits

Figure 1

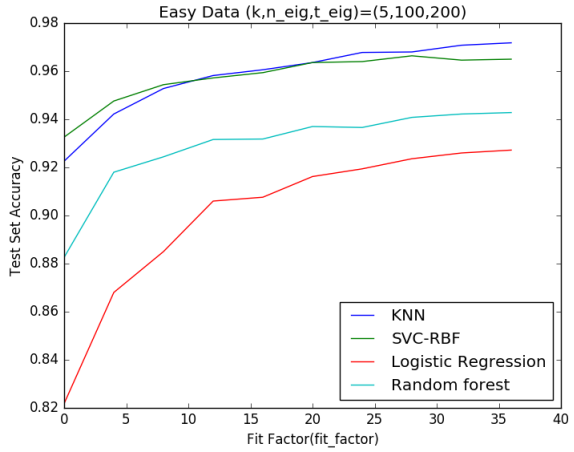
2. compute the mean(M) image of all the images in A and subtract it from each image.

3. Do the eigendecomposition of A^*A and sort the eigenvectors in decreasing order of eigenvalues.

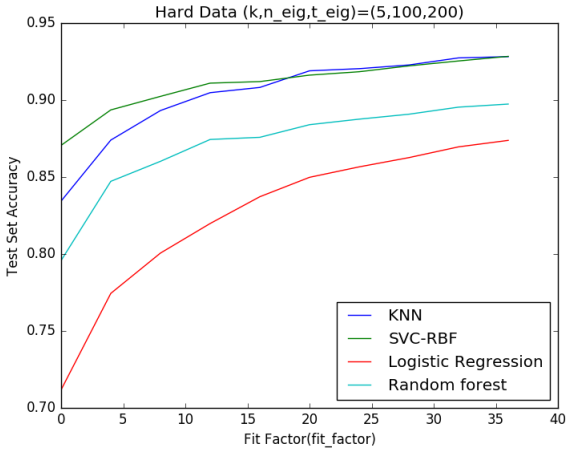
4. Normalize each of the eigenvectors to be a unit vector.

5. Subtract the mean image from all the training and test images, and then project each data to the t-dimensional eigenspace, by multiplying each sample row vector with the normalized eigenmatrix V

6. Take part of the projected training data, say $(= (fit\_factor)(t))$, and then train the KNN algorithm on it and test on projected test data.



(a) Accuracy vs Training Size-easy data



(b) Accuracy vs Training Size-hard data

Figure 2

5 Experiment

The first 16 eigendigits obtained from the covariance vector, can be seen in the Figure 1(d). It can be seen, that each eigendigit, represents the true digit in some way. For instance the 1^st eigendigit will have a higher activation for the digit 0. The $3^{rd}, 10^{th}$ and 12^{th} eigendigits are kind of blob detectors, which will have high activation for $0, 6, 8, 9$.

Now, The 3 dominant parameters in our lead algorithm, KNN, are :

- Number of nearest neighbors = k
- Number of training data for eigenspace construction = t_{eig}
- Number of training data used for training the model = $(fit\_factor)t_{eig}$

On running the PCA algorithm described in the previous section, with respect to each of the three parameters, we get the results as obtained in the Figure 1.
Now, it is quite visible that the accuracy of the PCA based KNN algorithm is close to 97% for easy test data, and close to 92% for the hard test data.
Though the algorithm does really good on the test data, in order to ensure robustness of the experiment it is important to compare it with other novel machine learning classification algorithm. The bank of algorithm, used to test against PCA based KNN algorithm are,

- Logistic Regression[5]
- Random Forest Classification[6]
- Support Vector Machines[7]

The results obtained using python's scikit-learn package[8] are shown in the the Figure 2. The results are quite promising as for a higher training data KNN with PCA seems to outperform SVC-rbf. The results will be properly explained in the next section, but it is quite

visible, that the algorithm is indeed very efficient.
Now in order to realize the fact, that the data information is lost while reducing the feature space, an experiment was performed, to reconstruct the original image from the lower dimensional feature space. This reconstruction, no matter how accurate our algorithm is, should not lead to a complete reconstruction, as information is lost in the feature reduction, which can't be recovered.
Steps:

1. A mean(M) subtracted set of 16 training images is taken and projected to $n_{eig}(= 100)$ dimensional eigenspace by post-multiplication of image by V.

2. The n_{eig} dimensional image is projected back to full 784 pixel image, by pre-multiplying the transpose of the projection vector with V and mean is added back to the image.

Now when the original image and reconstructed images were compared, there was some noise in the reconstructed image, evident in Figure 3.

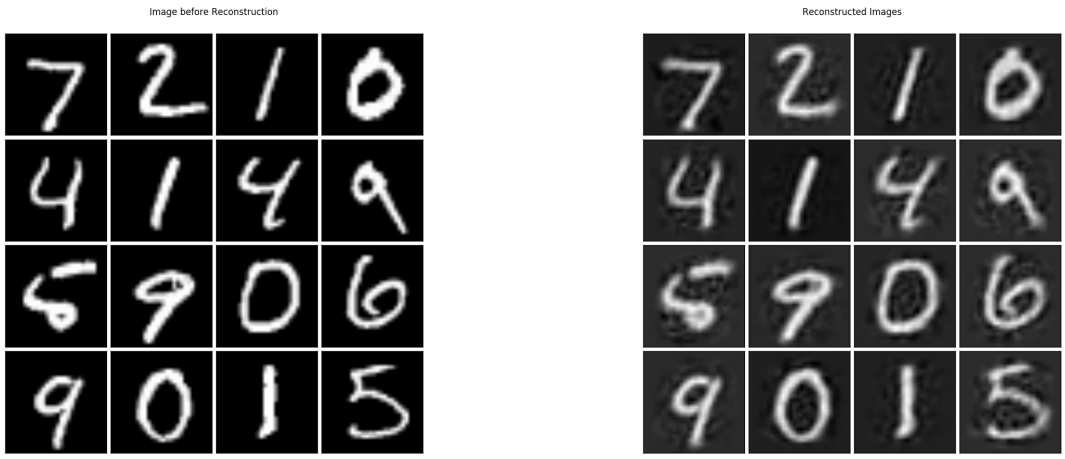
6 Observation & Justification

- The noise in the reconstructed image, from the reduced eigenspace image, clearly demonstrates the concept that some information about the sample data was lost in the compression. This aligns to our discussion in the concepts section. This also demonstrates the fact, that PCA is indeed a lossy compression.

- As the number of nearest neighbors used for KNN, increases, the accuracy more or less remains the same, with 5 neighbors being the best neighbour count, by a small factor. However, an important observation, is that the accuracy suddenly drops for k = 2, and then rises. This might be because, the algorithm has to randomly choose between two classes, in the event of the two neighbors belonging to different classes. This makes it worse, as now the algorithmm with the $P = \frac{1}{2}$ will choose the wrong class. This factor can be seen on all $k = even$, as one can observe a kink from the figure. It appears to be mostly due to tie-braking errors as neighbors are even.

- As the eigenspace dimension is increased, the probability is on the rising trend, as more the dimension of the projection more of the data information we are preserving, hence more robust is the model to the observed data, however, we can see that, after $t_{eig} = 250$, the improvement in accuracy tends to saturate. This validates, the fact, that most of the information of the data matrix, A is present in the top few singular values of A. Thus the top few eigenvectors of the covariance matrix contains most of the information of the data matrix A.

- As the training size(fit_factor x t_eig) for the algorithm(KNN, SVD, Random Forest, Logistic Regression) increases, the test set accuracy increases. However,this saturates very quickly, and providing more data, does not lead to significant improvement. This shows, that our model, has understood the representation of numbers w.r.t the eigendigits, quite reliably, with the minimum data possible, and is now able to generalize to different data input.



(a) Accuracy vs Training Size-easy data (b) Accuracy vs Training Size-hard data

Figure 3: Reconstruction for top n = 100 eigenvectors from 600 eigenvectors

- Our simple KNN algorithm with eigendigit based features, performs neck to neck with a very complex Support Vector Classification model with RBF kernel, which is good, as in Machine Learning more simple the algorithm, better the chances of generalization. This is called Occam's Razor.
- Logistic regression, being a linear model, was supposed to perform poorly, as the job is non-linear. But surprisingly, Random forest model, did better in terms of time of completion when compared to all other algorithms, which shows that it can be very useful, in an online, real-time Machine Learning setup.

7 Conclusion

The feature space reduction using eigenspace of the covariance of data, is a very robust method, and preserves most of the information, while tremendously reducing the runtime of the algorithm. The obtained test set accuracy of about 92% and 97% for easy and hard test set data using K-nearest neighbors algorithm, shows that the model has indeed learnt useful patterns from the rather low dimensional high level eigenspace features.

8 References

References

[1] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." Chemometrics and intelligent laboratory systems 2, no. 1-3 (1987): 37-52.

[2] Cover, Thomas M., and Peter E. Hart. "Nearest neighbor pattern classification." Information Theory, IEEE Transactions on 13, no. 1 (1967): 21-27.

[3] De Lathauwer, L., B. De Moor, J. Vandewalle, and Blind Source Separation by Higher-Order. "Singular value decomposition." In Proc. EUSIPCO-94, Edinburgh, Scotland, UK, vol. 1, pp. 175-178. 1994.

[4] Lanczos, Cornelius. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Los Angeles, CA: United States Governm. Press Office, 1950.

[5] Hosmer Jr, David W., and Stanley Lemeshow. Applied logistic regression.John Wiley and Sons, 2004

[6] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2, no. 3 (2002): 18-22.

[7] Schölkopf, Bernhard, Kah-Kay Sung, Chris JC Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik. "Comparing support vector machines with Gaussian kernels to radial basis function classifiers." Signal Processing, IEEE Transactions on 45, no. 11 (1997): 2758-2765.

[8] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.