Evaluating Methods of Handwritten Digit Recognition Using Principal Component Analysis

Connor Campbell – W18003255
Intelligent Systems KF5042

Abstract*— *As handwritten digit recognition (HDR) algorithms improve in accuracy and usage; questions arise as to the methods of doing so which will produce the best results. When using principal component analysis to extract features, comparing K-Nearest Neighbour's (KNN) and Support Vector Machine's (SVM) ability to classify digits, KNN comes out the more accurate algorithm at all analysed PCA component amounts, showing accuracies as high as 94.2% compared to SVM peaking at 90.9%.

I. Introduction

Data collection benefits from this, as handwritten papers can be automatically aggregated without the need for a human moderator to interpret the results. HDR can be broken down into two steps, interpretation followed by classification. The method of interpretation focused on in this paper is PCA however individual component analysis (ICA) [1] is another commonly used method. Once the data has been extracted via PCA this data is then classified using an algorithm giving the data a calculable value. In this paper two methods of classification are compared, K-Nearest Neighbour (KNN) and Support Vector Machine (SVM), to see which returns the most accurate results. Initially how the PCA algorithm functions will be discussed, followed by the workings of KNN and SVM along with how the paper plans to evaluate them. Results of the testing will then be discussed, concluding with final thoughts on the methods.

II. Feature extraction using PCA algorithm

The core functionality of PCA is to find clusters of data in a subspace and ensure these correspond to the maximum variance allowed by the test dataset [5]. The largest issue with this is, PCA becomes very sensitive to outliers due to minimising the mean square error highlighting anomalies and the high dimensionality of the data [6].

Once the data is standardized, principal components (PC) need to be identified, this is done by plotting out data using features as the axis. PCA begins by finding a line of best fit for this data. Take **PC1** as the line of best fit, $\mathbf{M_1}$ as the initial point, $\mathbf{M_2}$ as $\mathbf{M_1}$ scaled onto **PC1** and **O** as the origin. **PC1** is rotated until $\mathbf{M_2}$ shows the max distance from **O** possible. The right-angled $\mathbf{M_2}$, $\mathbf{M_1}$, **O** triangle ensures that as the line $\mathbf{M_2}$, **O** gets larger $\mathbf{M_2}$, $\mathbf{M_1}$ gets smaller ensuring **PC1** is the line of best fit. Due to this, **PC1** always represents the largest covariance, and the following PC will be the next largest.

From **PC1** a one-unit long vector is taken, this will be the eigenvector. The sum of squares of the distances from the points on **PC1** to **O** will be the eigenvalue. This process is then repeated, **PC2** will be the line perpendicular to **PC1** and **PC3** will be perpendicular to both **PC1** and **PC2.**

Dimensionality reduction is one of the main reasons PCA is used for feature analysis. Because of the nature of some data, including the dataset used in this paper, the number of dimensions being used can be too large. To deal with this, variance is measured among PCs to see which ones are the most important and these will be used as features of the dataset. In most cases, covariance matrices are used to measure the variance between two features [7].

III. Classification stage

A. Training and Testing

To initially prepare the algorithm for classification, a set of training data is needed; for this 1000 images will be used labelled with the number that they represent. This will allow the algorithm to have data to base its classification against. Typically, a greater volume of training data results in increased accuracy of testing. The remaining 100 will be used to test.

The main focus of the PCA will be searching for is shape, since features such as colour and texture will not be a factor in handwritten text. The shape is taken by looking at sections of the image and analysing them for patterns [8]. Once this is done for the training set, a set of classified points and principal components, are plotted based on the eigenvector returned during PCA. The number of PCA components being used during the analysis will begin at 80, increasing in increments of 10 until 120 components are used. The code used to analyse the methods is shown in Appendix 1.

B. K-Nearest Neighbour (KNN)

KNN uses a set amount, the K value, of closest points, to determine the classification of the point.

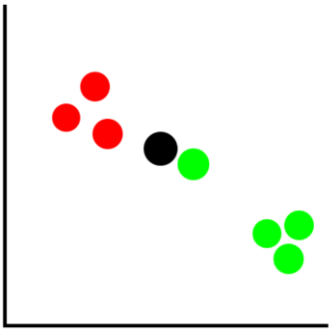

Figure 1 - KNN Classification

For example, in Figure 1, take the red points to have been classified as **X**, the green points have been classified as **Y** and the black point **P** is to be classified. If a K value of 3 was to be used **P** would be classified as **X** as of the 3 nearest points, 2 are **X** and 1 is **Y**. However, if a K value of 7 was used, **P** would be classified as **Y** as 3 are **X** and 4 are **Y.** To avoid ties, prime numbers are recommended for K values.

C. Support Vector Machines (SVM)

Using a support vector classifier (SVC), SVM classifies data based on which side of the SVC the data new data appears on. The SVC is a hyperplane that is calculated using kernel functions that identify the relationships between all of the existing data points [9].

IV. Results & Analysis

The results of the study can be found in the table shown in Table 1 and are graphed in Figure 2. The percentage is representative of the accuracy of the algorithm when classifying a digit. KNN showed a higher accuracy at every PCA level. When the number of components was raised from 80 to 90, the accuracy of both algorithms dropped, KNN by 0.3% and SVM by 0.5%. When this was raised again to 100, where KNN's accuracy remained at 93.5%, SVM's was raised by 0.8%. KNN's accuracy remained the same across 110 and 120 components, where at 110 SVM raised again to 90.9% and lowered to 90.2%. KNN showed its highest accuracy at 90 components and SVM showed its highest at 110 components. Both algorithms showed their lowest accuracy at 90 PCA components. KNN show a range of just 0.3% across all PCA component amounts where SVM shows a variance almost triple this at 0.8%

PCA Components	KNN	SVM
80	93.80%	90.60%
90	93.50%	90.10%
100	93.50%	90.80%
110	93.70%	90.90%
120	93.70%	90.20%

Table 1 - Accuracy shown by KNN and SVM during HDR (80-120).

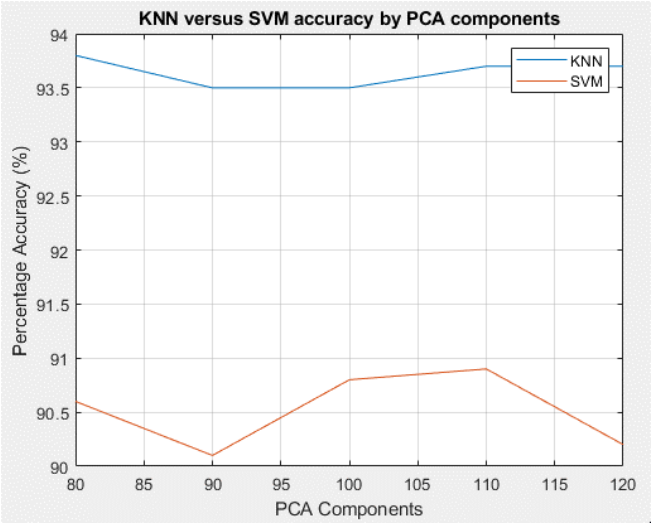

Figure 2 - KNN and SVN accuracy by PCA components (80-120).

It is shown by both of the algorithms not increasing in accuracy as the number of components increased that a higher volume of components does not result in greater accuracy. Because PCA is susceptible to outliers, the number of components increasing accuracy could be being offset by the number of outliers in the training set causing both KNN and SVM to misclassify digits. To account for this before PCA data could be standardised to ensure that outliers do not dominate datasets. This can be done on a small scale using the equation in Equation 1; however, it can be ordinated on a larger scale using more complex methods such as ORDIFLEX [4].

$$z = \frac{value - mean}{standard\ deviation}$$

Equation 1 - Data Standardization

Since the results appear to have plateaued, to get a greater understanding of the results shown the same test was run again, instead of at a much lower PCA level and rising starting at 3 PCA component up to 30 with increments of 3. Table 2 shows the results of this along with Figure 4 showing these results in a graph.

PCA Components	KNN	SVM
3	42.40%	48.40%
6	73.60%	71.80%
9	86.30%	81.90%
12	88.50%	84.40%
15	91.60%	85.50%
18	92.40%	87.20%
21	93.60%	89.90%
24	94.20%	89.50%
27	94.20%	90.60%
30	94.20%	90.70%

Table 2 - Accuracy shown by KNN and SVM during HDR (3-30).

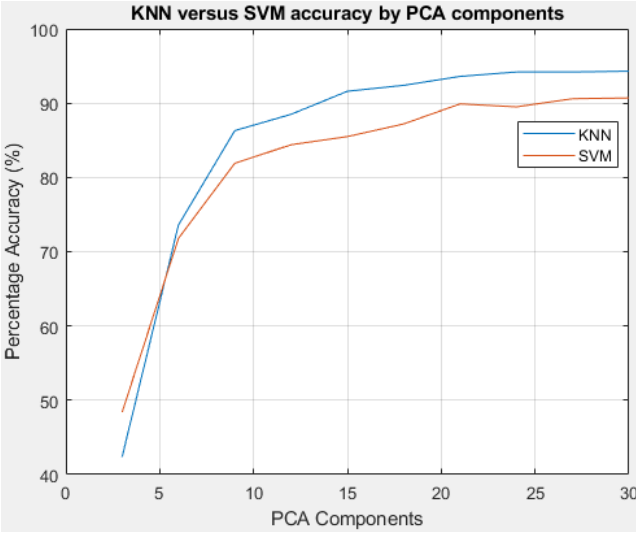

Figure 3 - Figure 2 - KNN and SVN accuracy by PCA components (3-30).

It can already be seen in Table 2 the accuracy of KNN reaching around the same accuracy at 21 PCA components as it did at 90 components, even yielding more accurate results at 30 components than it did at the range of 80 to 120 components. SVM appears to also be reaching around the same levels as its 80 PCA component classification at around 27 components, however, where KNN appears to be plateauing at around 24 components, SVM appears to still be increasing and could yield its highest accuracy within the 30 to 80 component range.

The greater accuracy shown by the lower PCA component amounts is likely a product of fewer features constructing fewer outliers.

V. Conclusion

When extracting features through PCA, simply using the highest amount of PCA components does not lead to the highest accuracy as PCA is subject heavily to outliers. When comparing KNN against SVM it is clear that across all tests KNN shows higher accuracies making it the clear choice of the two for the most precise HDR algorithm. However, neither method shows fully accurate results with KNN showing 5.8% and SVM showing 9.1% inaccuracy at their respective most accurate PCA component amounts, showing both algorithms still have much room to grow.

References

[1] B. Draper, K. Baek, M. Bartlett and J. Beveridge, "Recognizing faces with PCA and ICA", Computer Vision and Image Understanding, vol. 91, no. 1-2, pp. 115-137, 2003. Available: 10.1016/s1077-3142(03)00077-8.

[2] S. Roweis, "EM Algorithms for PCA and SPCA", Neural Information Processing Systems 10, pp. 1-3, 1998.

[3] Z. Jaadi, "A STEP-BY-STEP EXPLANATION OF PRINCIPAL COMPONENT ANALYSIS (PCA)", 2021.

[4] H. Gauch, "Noise Reduction By Eigenvector Ordinations", Ecology, vol. 63, no. 6, p. 2, 1982.

[5] A. Martinez and A. Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, p. 229, 2001. Available: 10.1109/34.908974.

[6] H. Kriegel, P. Kröger, E. Schubert and A. Zimek, "A General Framework for Increasing the Robustness of PCA-based Correlation Clustering Algorithms", pp. 1-4, 2008.

[7] S. Raschka, "Principal Component Analysis", 2015.

[8] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel , "Handwritten Digit Recognition with a Back-Propagation Network", pp. 3-6, 1990.

[9] W. Noble, "What is a support vector machine?", Nature Biotechnology, vol. 24, no. 12, pp. 1565-1567, 2006.

Appendix

Appendix 1 – Matlab Code

```
% clean
clear
close all
% Load dataset
load('MNIST_partitioned.mat');

% Shape the sets
trainSet =  reshape(trainImages1000, [784,10000])';
testSet = reshape(testImages100, 784, 1000)';
% Starting number of principal components
PC = 80;
iterations = 5;
increment = 10;
acc = [0 0; 0 0; 0 0; 0 0 ;0 0];

for i=1:iterations

% compute the mean digit
mu = mean(trainSet,1);
% centre the training data
trainSet = bsxfun(@minus,trainSet,mu);
% Generate eigendigits based on the number of prinicpal components being
% used.
coeff = pca(trainSet);
eigendigit = coeff(:,1:PC);

% generate the training features
trainFeatures = eigendigit'*trainSet';

%% Subspace projection
% centre the test set
testSet = bsxfun(@minus,testSet,mu);
% subspace projection
testFeatures = eigendigit'*testSet';

%% Build a KNN classifier.
% Parameters are the training features and their respective labels
mdl = fitcknn(trainFeatures',trainLabels1000);

% predict the similarity using the classifier and the test featuresas parameters.
labels = predict(mdl,testFeatures');

% find the images that were recognised and their respective labels
```

```
correctRec = find(testLabels100 == labels');

% compute and display the recognition rate
KNNresult = length(correctRec)/length(testLabels100)*100;
acc(i,1) = KNNresult;

%% Build a SVM classifier.
% Parameters are the training features and their respective labels
mdl = fitcecoc(trainFeatures',trainLabels1000);

% predict the similarity using the classifier and the test features as parameters.
labels = predict(mdl,testFeatures');

% find the images that were recognised and their respective labels
correctRec = find(testLabels100 == labels');

% compute and display the recognition rate
SVMresult = length(correctRec)/length(testLabels100)*100;
acc(i,2) = SVMresult;

fprintf('----------- PCA at: %0.3f ----------- \n',PC);
fprintf('The recognition rate of KNN is: %0.3f \n',KNNresult);
fprintf('The recognition rate of SVM is: %0.3f \n',SVMresult);
% Add increment
PC = PC + increment;
end

pca = [80; 90; 100; 110; 120];

figure;
plot(pca,acc);

title('KNN versus SVM accuracy by PCA components');
xlabel('PCA Components');
ylabel('Percentage Accuracy (%)');
grid on

legend('KNN','SVM');
```