Devon Callanan
Machine Learning
Homework 3

PCA

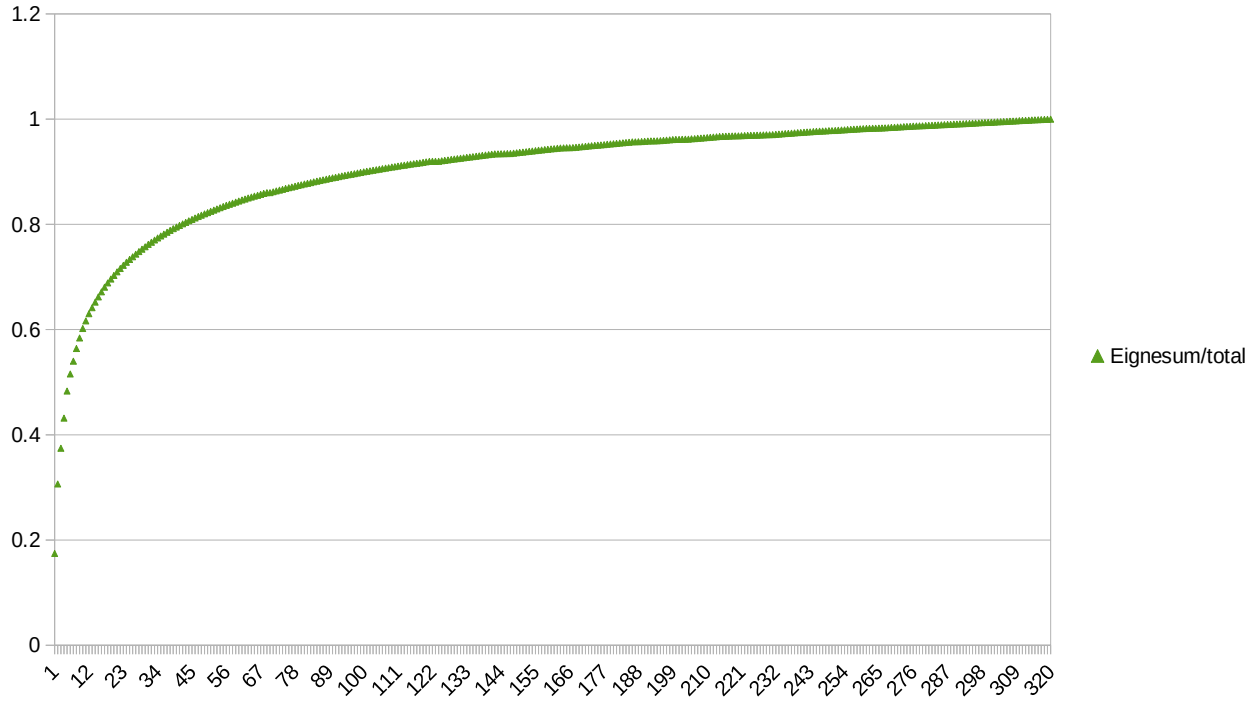
Principle component analysis was employed in part one for dimensional reduction of the image data. First data was centered by subtracting the mean of the training set. Then, the eigenvalues of the matrix L = transpose(A)*A were computed and multiplied by the matrix A to get eigenvectors of the convariance matrix through a simple workaround.

Use $L=A^TA$ instead of $\Sigma=AA^T$
If $\mathbf{v}$ is eigenvector of L
then $A\mathbf{v}$ is eigenvector of Σ

Proof:
$$L\,\mathbf{v} = \gamma\,\mathbf{v}$$
$$A^TA\,\mathbf{v} = \gamma\,\mathbf{v}$$
$$A\,(A^TA\,\mathbf{v}) = A(\gamma\,\mathbf{v}) = \gamma\,A\mathbf{v}$$
$$(A\,A^T)A\,\mathbf{v} = \gamma\,(A\mathbf{v})$$
$$\Sigma\,(A\mathbf{v}) = \gamma\,(A\mathbf{v})$$

Figure 1: Clever workaround and proof

It was found that making these resulting eigenvectors or unit length my dividing by the magnitude improved accuracy. These eigenvectors of the covariance matrix are considered the *eigenfaces* and represent a new lower dimensional space. Since any image can be seen as a linear combination of these eigenfaces, only the dot product of the input image against the eigenface space is needed to represent the image. The chart bellow show the contribution to the image each eigenvector has by plotting the eignevalues.



Since most of the contribution is made using the first 130 or so egencectors, we do not need to project along the remaining ones, further reducing the dimensionality.

The accuracy using five fold cross validation of the PCA with full image size (112,92) and dimensionality of 130 is shown bellow. 1NN is used to make classifications, finding the closes point with a euclidean distance.

pic dims	10304
Dimensionality	130
Acc 1	0.9875
Acc 2	0.975
Acc 3	0.9875
Acc 4	0.9875
Acc 5	0.9625
acc avg	0.98

By halving the image dimensions (56,46) we we have less data to work with, however we attain similar results.

pic dims	2576
Dimensionality	130
Acc 1	0.9875
Acc 2	0.975
Acc 3	0.9875
Acc 4	0.9875
Acc 5	0.9625
acc avg	0.98

LDA

Linear discriminant analysis is another avenue for dimensionality reduction. LDA strives to project features into a space where dimensions represent the maximum cluster separation. First, the means of each class are computed then the within class scatter:

$$S_w = \sum_{i=1}^{c} S_i$$

$$S_i = \sum_{X \in D_i} (X - m_i)(X - m_i)^t$$

$$m_i = \frac{1}{n_i} \sum_{X \in D_i} X$$

Figure 2: Within class scatter for a multi class LDA

Then the between class scatter is calculated:

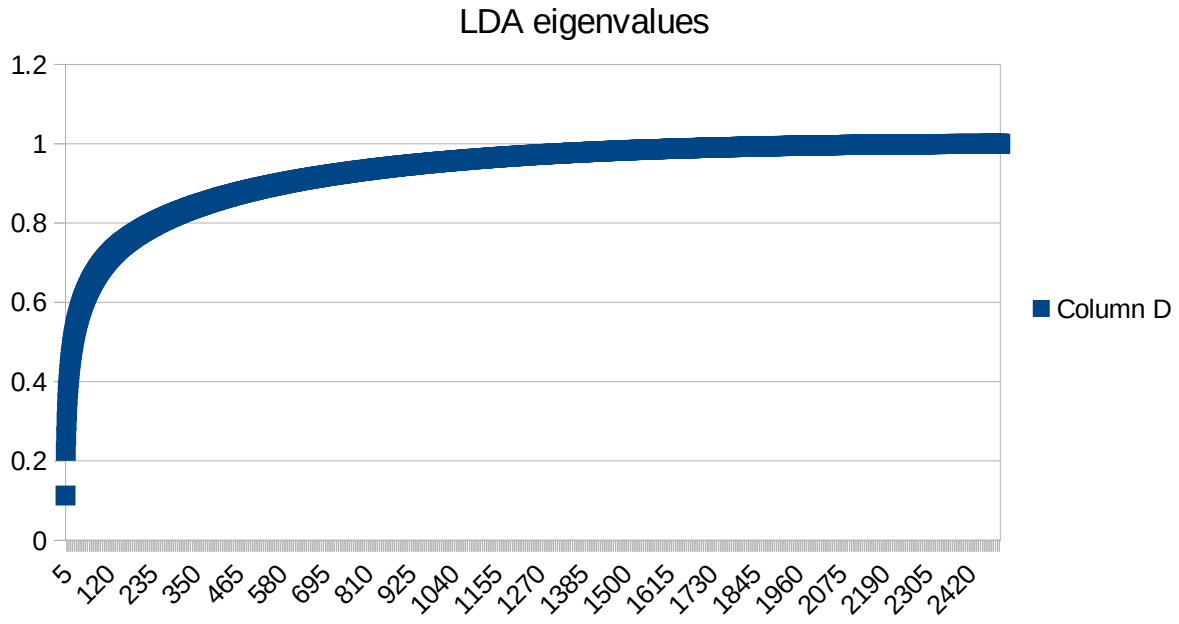
$$S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^t$$

Figure 3: Between class scatter

Finally the eigen problem shown bellow is solved creating the new dimensionality. As with PCA, some of these eigenvectors provide more variance and so only a limited number of the most potent are needed. The chart bellow shows that after 300 dimensions, there is little else to gain.

$$\boldsymbol{S_W^{-1} S_B v} = \lambda \boldsymbol{v}$$

Figure 4: Eigen problem



Using only these dimensions, and the same 1nn distance classification, the following accuracies are reported.

pic dims	10304
Dimensional ity	300
Acc 1	0.95
Acc 2	0.95
Acc 3	0.975
Acc 4	0.9625
Acc 5	0.9375
acc avg	0.955

By halving the image dimensions (56,46) we we have less data to work with, however we attain similar results.

pic dims	2576
Dimensional ity	300
Acc 1	0.975
Acc 2	0.9625
Acc 3	0.9875
Acc 4	0.9625
Acc 5	0.9375
acc avg	0.965

Combined

By combining PCA and LDA we can further reduce the dimensionality of the feature space. For PCA, 90 dimensions were used and the resulting vectors were passed into an LDA pipeline with 60 dimensions.

pic dims	10304
Dimensionality	90, 60
Acc 1	0.95
Acc 2	0.9
Acc 3	0.9875
Acc 4	0.925
Acc 5	0.9625
acc avg	0.945

Usage

First install dependencies listed in requirements.txt (preferably in a virtual environment), then run ‘python knn.py [pca lda both] cut1 cut2’. The first arg is the type of dimensionality reduction to run, the cuts are the dimensionality to reduce to. Cut2 is only used in the ‘both’ model. Image size is manually tuned in the DataSet class load() method, see line 29.

Example:

python knn.py pca 60 → use pca to reduce the images to a feature space of 60 before performing 1nn.