TP3: PCA and kNN

by Martino Ferrari

1 PCA

Due to performance problem of my computer in combination with Octave I used 500 samples instead of 5000 with 1 as step size for exploring m. More over the PCA function is not implemented in Octave so I implemented my self.

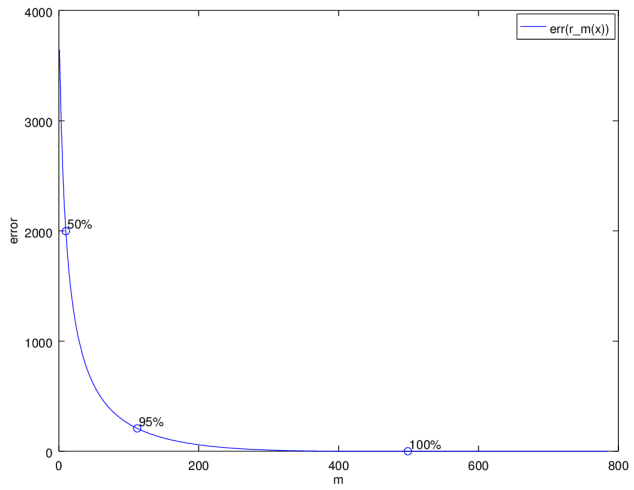

Figure 1. Error of $\mathcal{X}$

In figure 1 is possible to see the evolution of the error measured as the square error of $\mathcal{X}$ and its reconstruction $r_m(\mathcal{X})$ within the complete range of m (in this case $1, ..., 784$).

The accuracy of 50% is reached with 10 components while the 95% with 112 components and 100% within 499 components.

The 100% accuracy with only 499 components means that the 285 resting components are always empty, meaning that this 285 pixels are empty and never used. However is possible that with 5000 training image this value could be bigger.

Figure 2. 5 samples reconstructed

In figure 2 I reconstructed 5 samples using different number of m components of the PCA. It's possible to see that with very few components the reconstruction is already significant.

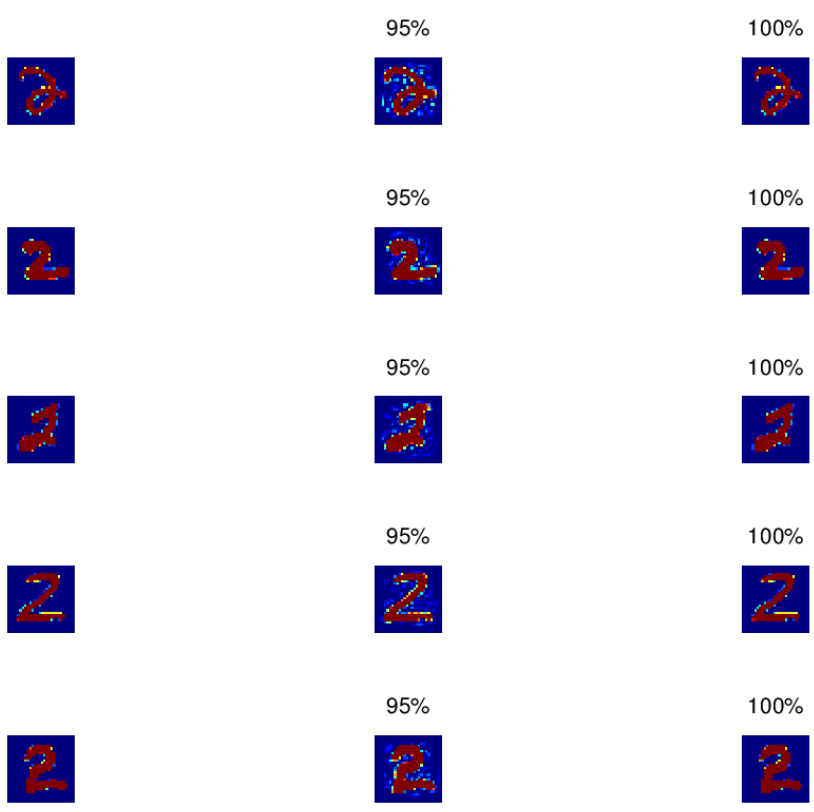

Figure 3. High m reconstruction

The reasonable value of k depends of the use of the reconstructed images, for compression and visualization a high number of components (as shown in figure 3), close to the 100% precision (499 components), for data analysis purpose is enough a much smaller number of components, bigger then 10 (that has only 50% of precision) but much lower of the 112 (95% of precision), in this way the noise will be reduced and only the more significant part of the image will be analysed.

2 *k*NN classification

A 1-NN classificator is very simple, it search the training samples more similar (with the smaller difference) to the samples and classify the sample as the selected training sample.

With the training data and a simple 1-NN classifier implemented by classifying 500 test samples has precision of 97.2%.

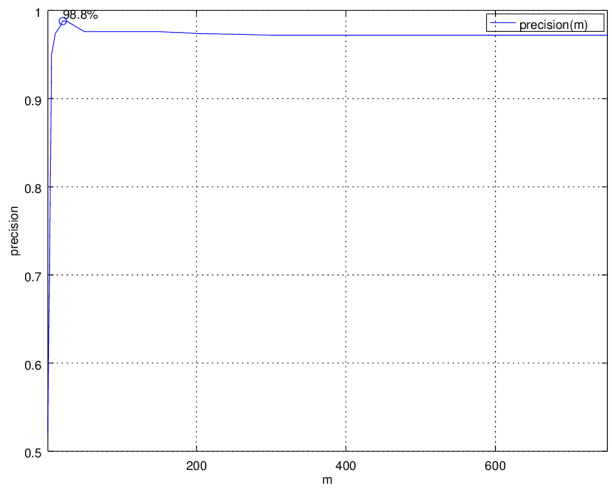

Figure 4. Precision of the 1-NN classificier with the PCA in function of m

Using this classifier in combination with the PCA permits to reach a precision close to 99% within the range of 21 to 26 components as shown in the figure 4.

In figure 4 is possible to observe how the precision of the 1-NN classifier depends on the number m of components used (as expected in the previous exercise), it could be natural to think that the best precision is reached whit high number of components (or using the real test image without any PCA). However in reality this is not true, in fact the relevant information is carried by the firsts components of the image and adding to many components add noise to the information.

Using a correct number of components (not to few and not to much, in this particular case around 25) give all the information needed for the classification without additional noise.

In conclusion it would be possible to classify with very high precision hand written numbers using only 25 representative pixels (instead of 784).

3 Code

As already told, the code was developed for Octave, and never tested with Matlab (and I suspect some operation are not allowed in Matlab as the subtraction of a matrix and a vector). However Octave is free and avaible for all platform.

→ pca.m	return the pca, eigenvalues and normalized eigenvalues
→ nkk.m	return the index of the
→ rm.m	reconstruct an image using m pca components
→ plot_e1.m	plot the results of the ex. 1
→ find_m.m	return m components given a certain precision
→ e1.m	first part of the ex. 1
→ e1-2.m	second part of the ex. 1
→ e2.m	execute code for ex.2