Cogs 185: Homework 4
Michael Hatch
A09910883

Introduction:

Principal components analysis (PCA) is a common method of dealing with large amounts of data. In PCA (as in other dimensionality reduction techniques) the goal is to encode enough information about a signal or observation to accurately characterize it in a new feature space that is (ideally much) smaller than the original space.

PCA is notoriously sensitive to outliers (for example, different lighting patterns or the presence of sunglass on an image of a face), and thus outlier-resistant methods of PCA have become an attractive idea. Robust PCA (RPCA) works by isolating deviations in a feature set in an error matrix and making that matrix as sparce as possible.

Generally used as an exploratory or visualization technique, PCA can also be useful for feature extraction. Taking every principle component of a dataset will provide all of the information as the original dataset, but generally the last principle components contain mostly noise. Due to this, removing them can actually boost the effectiveness of classifiers, as it allows them to identify and learn relevant features (like complexion, shape of the eyes and mouth, etc) by throwing out information that is not useful (imperfections due to image resolution, small differences in light/shading, etc).

To test the effectiveness of feature extraction, and the effectiveness of RPCA over PCA, we will be performing both PCA and RPCA on a dataset of 165 32x32 pixel faces. In addition, to test whether RPCA is effectively excising error from the reduced representation spaces, we will also be performing PCA on the A matrix produced from RPCA.

Methods:

Computation was performed on the Yale expressions dataset, found here. There are 11 images each of 15 different individuals, each image with a different expression or configuration (ie wearing sunglasses). Code can be found here.

The principal components of the dataset were calculated using the standard covariance matrix algorithm. RPCA was also performed via the Proximal gradient algorithm found here. In addition PCA was performed on the A matrix computed with RPCA for feature engineering purposes.

Datasets were collected using matlab, code here. From there, data was imported into Python and classified using Random Forests. Methodology has been described in detail on previous reports, for example see here.

Results and discussion:

The results are shown on the table below. The accuracy for random forests performed on each transformed dataset are displayed, and in parenthesis is the number of decision trees used by the forest. Chance accuracy for 15 classes is approximately 6%.

	Full dataset (1024 components)	165 components	22 components	5 components
PCA		33.33% (162)	75.76% (109)	36.36% (67)
RPCA	72.72% (54)	42.42% (27)	42.42% (7)	33.33% (137)
RPCA -> PCA		39.39% (76)	69.67% (142)	63.64% (31)

The results are generally as expected. Classifying on the full A matrix of the RPCA datase is quite effective, which should be very close to classifying on the original dataset, since the only information that has been removed is irrelevant to the task of determining which face is being shown.

RPCA is also more effective than normal PCA on a 165 components (the same number of components as observations) basis. Doing a further PCA on the A matrix decreases accuracy slightly, which could be due to the fact that the 165-component RPCA basis still encodes the majority of the relevant information, and transforming it again hurts accuracy more than it helps.

The 22 component basis generally shows an improvement in the classifying power, with the exception of RPCA, and it is here that performing PCA on the A matrix is shown to be an effective technique. Normal PCA does better because most of the components being removed represent noise, leaving the relevant information to be used with training. RPCA isn't improving since the error was removed with the initial RPCA process; the information being lost is likely relevant for the classifier, but only a small portion of that information is being lost (since the first 22 components represent most of the relevant information), so there is no decrease in activity either. The principal components of the A matrix are serving as a further filter for relevance, the variance explained is more concentrated in the first few components (since there are a maximum of 165 components as opposed to 1024); because of this less relevant information is lost by shrinking the double-processed matrix than the A matrix.

This also explains why Sample 3 also performs fairly well even classifying on only 5 components. Both PCA and RPCA are losing too much relevant information by throwing out so many principal components, that by this point it is hurting their classification ability, although RPCA isn't as negatively impacted as normal PCA.

There are some anomalous results, in particular normal PCA with 22 components really should not be more effective than RPCA on 1024 components, and in general the 22-component sets should not be as superior to their 165-component counterparts as they are.

The most likely culprit for this behavior is the smaller size of the dataset - which can directly impact the classifier, educes the usefulness of PCA as a reducer of dimensionality, and makes the accuracy more susceptible to smaller stochastic changes in the test/training split of the data (if 7 of the 11 images of one individual find their way into a testing set, for example).

However, it’s also possible that unexpected results are the product of the nature of the classifier and the cross-validation process, as different numbers of trees in each forest could be driving the success of the classifier more than the data itself.