A comparison of PCA, KPCA and ICA for dimentionality reduction

I. Problem addressed and its Importance

PCA (Principal components analysis), KPCA (Kernel based Principal components analysis), ICA (Independent components analysis) are important feature extraction techniques used for dimensionality reduction. It is important to reduce the number of dimensions under consideration as they increase the computation time heavily and we might also have irrelevant features in the data collected. Dimensionality reduction helps reduce the curse of dimensionality and cuts the computation cost. These preprocessing steps can also help in reducing the noise, making the data easier to work with as they create uncorrelated features of the data. Reducing the number of dimensions in the dataset helps in deeper understanding of the problem. [1]

PCA linearly transforms the original inputs into new uncorrelated features. KPCA is a nonlinear PCA developed by using the kernel method. In ICA, the original inputs are linearly transformed into features which are mutually statistically independent.

We want to analyze and gain deeper understanding of these algorithms for high dimensional data and learn the differences in the performance between these. For this purpose we compare all the three techniques on several classification problems and analyze the results.

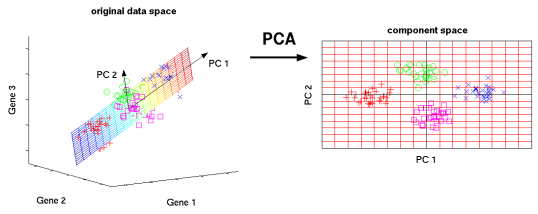

Fig. 1. Image from [2] showing the conversion of 3 dimensional gene expression data which can be represented in 2 dimension using PCA.

II. Related work in this area

The paper, [3] compares PCA, KPCA and ICA for three different datasets that include Sunspot data, Santa Fe data and price of 5 futures contracts. It compares the techniques for classification problems using SVM. Performance of feature extraction algorithm based preprocessing is compared between themselves and with a case where no preprocessing has been done. And it finds the results that KPCA performs better then ICA which is followed by PCA and all three cases do better then the case where no preprocessing has been done. One of the deficiencies of the paper is that they compare the algorithm only on SVM based learning and generalize the performance based on that. It would be better to compare the performance on some other learner as well to see the consistency in results. Also in KPCA the paper only uses gaussian kernels while using other kernel functions might also improve the results in specific problems.

[4] discusses most of the algorithms in a theoretical format and with the tests conducted on ideal datasets created artificially. This performance might vary on real life problems that have a lot of factors influencing the training and testing data, specially as it contains noise, which is not accounted for in the discussions of the paper. While [1] talks about the algorithmic implementations and gives an overview of the concepts without going into mathematical details.

III. Approach

A. Framework

We used python to implement all the learning and dimensionality reduction algorithms which include PCA, KPCA, SVM etc. Inside python we used Numpy and Scipy libraries for mathematical operations such as matrix operations, singular value decomposition and eigenvalue calculations. We got a huge support from the sklearn package which provides learning algorithms, kernel function processing and also contains machine learning datasets such as digits. We implemented the three dimensionality reduction algorithms and compared performance on three datasets for handwritten digit recognition, face recognition and arrhythmia detection. This was done for both SVM and decision tree for comparison.

Our implementation of principle components analysis (PCA) uses Numpy to calculate the singular value decomposition (SVD) of feature matrix. In our implementation of kernel based principle components analysis (KPCA) we use Numpy to calculate eigenvalues and sklearn to map features into kernel space. It must be noted that we do not use the algorithms that are built inside sklearn and rather have implemented on our own. And only use limited functions from the libraries.

B. Principle Components Analysis

The main idea behind principle components is that the direction of the principle component is the direction of maximum variance in the data. The algorithm first centers the data by subtracting the mean and then chooses the direction with largest variance as the principle component. It then looks in the variance that remains and places an axis in the direction that is orthogonal to the first axis and has largest variance. It then does this iteratively and creates orthogonal basis for the dataset. We can ignore the axis with very little variance as they don't have a lot of impact on the prediction of the learner and might also be only noise.

In mathematical terms, by calculating the eigenvectors of the covariance matrix of the original inputs, PCA linearly transforms a high-dimensional data into a lower dimensional space which are uncorrelated and orthogonal. We first compute the eigenvalues of the feature matrix, then sort the eigenvalues and ignore the really small values. Then we transform our data into the eigenspace formed by the selected eigenvectors.

C. Kernel based Principle Components Analysis

Kernel based principle components analysis is a non linear PCA created using the kernel trick. KPCA maps the original inputs into a high dimensional feature space using a kernel method.

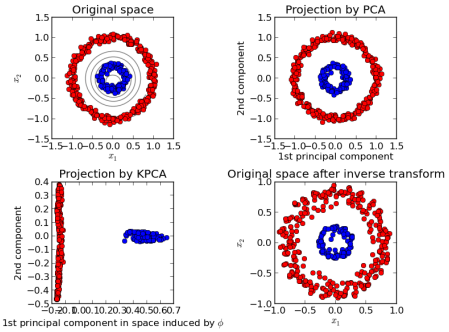

Fig. 2. Image seen in sklearn KPCA demo also specified in [1]. Created with our code, it shows the difference between PCA and KPCA.

Mathematically, we transform the current features into an high-dimensional space and the calculate eigenvectors in this space. We ignore the vectors with really low eigenvalues and then do learning in this transformed space. KPCA is computationally intensive and takes a lot more time compared to PCA. The reason being that the number of training data points in KPCA is much higher then PCA. So number of principle components that need to be estimated is also much larger.

D. Independent Components Analysis

The common case of ICA is the blind source separation problem where the goal is to recover mutually independent but unknown source signals from their linear mixtures without knowing the mixing coefficients. Two differences between PCA and ICA are that the components here are statistically independent and not uncorrelated. And second, the un-mixing matrix is not orthogonal like PCA [3].

The algorithm works on the principle of minimizing mutual information between the variables, minimizing mutual information is the correct criteria for judging independence. Also minimizing mutual information is same as maximizing entropy. There are several algorithm for doing ICA one of the most popular ones being FastICA [1].

E. Support Vector Machines

Support vector machines are supervised learning algorithms used for classification and regression analysis. A SVM constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class, since in general the larger the margin the lower the generalization error of the classifier. SVMs are good at classification because they minimize generalization error rather then training error [3].

F. Decision Trees

The idea of decision tree is that we break the classification down into a set of choices about each feature in turn, starting at the root of the tree and progressing down the leaves in an if-else ladder form. To create the tree we decide the ordering of the features based on information gain, and keep the feature with high gain at the top of the tree.

G. Parameter Selection

One of the issues with the pre-processing algorithms is that we need to select the number of principle components that need to be used. We start with the first principle component and keep increasing the number till the cross validation error reduces. So that we only compare the best results between the algorithms. For KPCA we try different kernel functions such as linear, radial basis function and polynomial and the best result is used for comparison. The next section discusses the results of our analysis.

IV. Evaluation

We use the misclassification error as the criteria for comparison of various algorithms, which is ratio of wrong classifications to the total number of classifications. We split the training set into 10 different groups and use 9 for training and 1 for the testing. This is repeated until every group has been used as the validation set once. This is also done for the case where none of the algorithms have been applied to the data.

A. Handwritten Digits Dataset

The dataset used is the Semeion Handwritten digit dataset from the UCI machine learning repository [5] . The results observed are presented in the table below.The dataset consists of handwritten digits of 80 people for all digits 0-9 and was scaled to a 8x8 image so 64 values of 1 or 0. The results for comparing the three algorithms is presented below .

	SVM		Decision Tree	
Processing	Dimensions	Error	Dimensions	Error
Original	–	42.2%	–	34.55%
PCA	3	15.5%	10	14.79%
KPCA	300	2.1%	80	13.35%
ICA	18	4.2%	12	13.87%

B. Face Recognition Dataset

The dataset used in the experiment is the Wild Face recognition dataset from University of Massachusetts. The dataset has 1850 attributes. There are 1680 people in the database, making it difficult to classify into different classes.

Hence, we filtered the images on the criteria that there should be atleast 70 images of the person in the analysis. Most images have only a few examples, while some have a lot of examples, which creates an imbalance in the data and increases error. Then we resized the images to a 0.4 size so that computation was faster.

	SVM		Decision Tree	
Processing	Dimensions	Error	Dimensions	Error
Original	–	58.82%	–	41.69%
PCA	120	13.95%	100	13.96%
KPCA	280	16.32%	260	17.46%
ICA	210	16.48%	200	13.65%

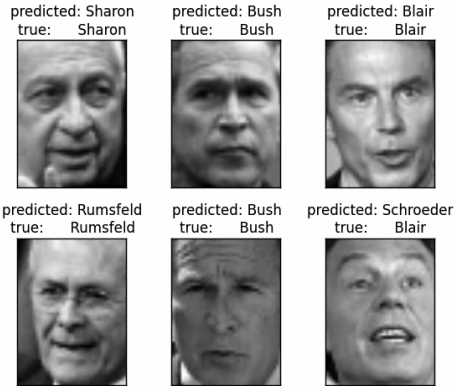

Fig. 3. Image showing the results of the Face Recognition using PCA.

C. Arrhythmia Dataset

The aim of the dataset to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 16 groups [6]. The data has 279 attributes, and then the corresponding target vectors. The attributes consist of Age, Sex, Height, Weight, Heart rate and various other medical measurements.

	SVM		Decision Tree	
Processing	Dimensions	Error	Dimensions	Error
Original	–	46.4%	–	47.33%
PCA	60	32.2%	70	35.25%
KPCA	200	26.1%	180	23.96%
ICA	75	30.7%	90	29.22%

V. Discussion

In the Handwriting based digit recognition the best results are obtained by KPCA, followed by ICA and then PCA. Further these results are much better then the case when we use the learner without any pre-processing. This demonstrates the fact that dimensionality reduction can improve the classification error along with cutting the computation time of the learner. This is in coherence with the results discussed in [3]. Though we see the best results are obtained using KPCA but that has a lot more principle components compared to PCA and ICA. This is mainly due to the filtering of unnecessarily attributes being part of the analysis and noise being removed from the handwriting.

For the Face recognition test we see the best results being generated by PCA, followed by ICA and then by KPCA. But these outperform the learner without any pre-processing by a huge margin. The main reason being that the original learner had 1850 attributes with most of the data as useless noise due to image being clicked outside and from various angles. So the learning isn't exactly out of images. PCA filters out important characteristics of the face that can be best used in the classification of the images and the learners performance is improved dramatically due to that reason.

In the Arrhythmia classification problem the results are not that great in comparison to the previous results but we still observe that KPCA is better then ICA which is followed by PCA. But still we find that pre-processing of the data improves the performance of the learner. From all the three cases we actually find that preprocessing has improved the results of the classification problem. But the problem with using KPCA is the huge number of dimensions, that is because generally the number of training instances is more then the number of attributes. So KPCA has to look for much more principle components(# instances) compared to PCA and ICA(# attributes). KPCA and ICA perform better then PCA as they can explore more higher order information then PCA. Instead of the covariance matrix used in PCA, entropy in ICA could take into account the higher order information of the original inputs.

In our tests we calculate Eigenvalues from covariance matrix instead of SVD as SVD is difficult to perform with the current computing resources and that can further improve the results. Also in the application of SVM and Decision trees can be further improved and optimized to yield better results. One interesting extension for the work can be the optimization and improvement of KPCA to reduce computation time with using only selected data rather then all the training data as the dimensions increase highly when we use the complete training data.

It is important to note that the objective is to compare the different algorithms and not find the best algorithm for the tasks compared. So the results aren't the best face recognition algorithm rather a comparison of three pre-processing steps keeping other settings constant.

References

[1] S. Marsland, *Machine Learning : An Algorithmic Perspective*. CRC Press, 2009.
[2] M. Scholz, *Approaches to analyse and interpret biological profile data*. PhD thesis, 2006.
[3] L. Cao, K. Chua, W. Chong, H. Lee, and Q. Gu, "A comparison of pca, kpca and ica for dimensionality reduction in support vector machine," *Neurocomputing*, vol. 55, pp. 321–336, 2003.
[4] L. van der Maaten, "Dimensionality reduction: A review," ticc, Maastricht University, May 2009.
[5] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.
[6] H. Guvenir, "A supervised machine learning algorithm for arrhythmia analysis," in *Computers in Cardiology 1997*, pp. 433 –436, sep 1997.
[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[8] J. Shlens, "A tutorial on principle components analysis," 2003.
[9] J. Stone, "Independent components analysis: A tutorial introduction," 2004.
[10] F. Pedregosa, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[11] G. B. Huang, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," 2007.