Experiment No.7: Principal Component Analysis (PCA) for Increasing Speed of ML Algorithm

Aim- To study, understand and implement a PCA algorithm.

Theory-

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables.

Each of the principal components is chosen in such a way so that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components, the first principal component has a maximum variance.

** **About Dataset-** Title: MNIST database of handwritten digits

This has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.
The images of the downloaded dataset are contained in mnist.data and have a shape of (70000, 784) meaning there are 70,000 images with 784 dimensions (784 features). The labels (the integers 0–9) are contained in mnist.target. The features are 784 dimensional (28 x 28 images) and the labels are simply numbers from 0–9.

Code -

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')
```

```
from sklearn.model_selection import train_test_split
# test_size: what proportion of original data is used for test set
train_img, test_img, train_lbl, test_lbl = train_test_split( mnist.data, mnist.target, test_size=1/7.0, random_state=0)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Fit on training set only.
scaler.fit(train_img)
# Apply transform to both the training set and the test set.
train_img = scaler.transform(train_img)
test_img = scaler.transform(test_img)
from sklearn.decomposition import PCA
# Make an instance of the Model
pca = PCA(.95)
pca.fit(train_img)
train_img = pca.transform(train_img)
test_img = pca.transform(test_img)
from sklearn.linear_model import LogisticRegression
# all parameters not specified are set to their defaults
# default solver is incredibly slow which is why it was changed to 'lbfgs'
logisticRegr = LogisticRegression(solver = 'lbfgs')
logisticRegr.fit(train_img, train_lbl)
# Predict for One Observation (image)
logisticRegr.predict(test_img[0].reshape(1,-1))
# Predict for Multiple Observations (images)
logisticRegr.predict(test_img[0:10])
logisticRegr.score(test_img, test_lbl)
```

Results

```
from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784')

from sklearn.model_selection import train_test_split
# test_size: what proportion of original data is used for test set
train_img, test_img, train_lbl, test_lbl = train_test_split( mnist.data, mnist.target, test_size=1/7.0, random_state=0)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Fit on training set only.
scaler.fit(train_img)
# Apply transform to both the training set and the test set.
train_img = scaler.transform(train_img)
test_img = scaler.transform(test_img)
from sklearn.decomposition import PCA
# Make an instance of the Model
pca = PCA(.95)
pca.fit(train_img)
train_img = pca.transform(train_img)
test_img = pca.transform(test_img)
from sklearn.linear_model import LogisticRegression
# all parameters not specified are set to their defaults
# default solver is incredibly slow which is why it was changed to 'lbfgs'
logisticRegr = LogisticRegression(solver = 'lbfgs')
logisticRegr.fit(train_img, train_lbl)
# Predict for One Observation (image)
logisticRegr.predict(test_img[0].reshape(1,-1))
# Predict for Multiple Observations (images)
logisticRegr.predict(test_img[0:10])
logisticRegr.score(test_img, test_lbl)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
0.9201
```

Discussion-