Module1

May 23, 2022

1 Homework 1: PCA

1.1 Problem 1 - Principal Component Analysis

In this problem you'll be implementing Dimensionality reduction using Principal Component Analysis technique.

The gist of PCA Algorithm to compute principal components is follows: - Calculate the covariance matrix X of data points. - Calculate eigenvectors and corresponding eigenvalues. - Sort the eigenvectors according to their eigenvalues in decreasing order. - Choose first k eigenvectors which satisfies target explained variance. - Transform the original data of shape m observations times n features into m observations times k selected features.

The skeleton for the *PCA* class is below. Scroll down to find more information about your tasks.

[1]:
```
import math
import pickle
import gzip
import numpy as np
import pandas
import matplotlib.pylab as plt
%matplotlib inline
```

[3]:
```
from sklearn.preprocessing import StandardScaler
```

[54]:
```
class PCA:
    def __init__(self, target_explained_variance=None):
        """
        explained_variance: float, the target level of explained variance
        """
        self.target_explained_variance = target_explained_variance
        self.feature_size = -1

    def standardize(self, X):
        """
        standardize features using standard scaler
        :param X: input data with shape m (# of observations) X n (# of␣
↪features)
```

```
        :return: standardized features (Hint: use skleanr's StandardScaler.␣
↪Import any library as needed)
        """
        # your code here
        scaler = StandardScaler()
        return scaler.fit_transform(X)


    def compute_mean_vector(self, X_std):
        """
        compute mean vector
        :param X_std: transformed data
        :return n X 1 matrix: mean vector
        """
        # your code here
        data = self.standardize(X_std)

        return data.mean(axis=0)



    def compute_cov(self, X_std, mean_vec):
        """
        Covariance using mean, (don't use any numpy.cov)
        :param X_std:
        :param mean_vec:
        :return n X n matrix:: covariance matrix
        """
        # your code here
        n,m = X_std.shape
        cov_matrix = np.empty((m,m))

        for i in range(m):
            for j in range(m):
                cov_matrix[i,j] = np.sum((X_std[:, i] - mean_vec[i]) * (X_std[:
↪, j] - mean_vec[j])) / (n-1)

        return cov_matrix



    def compute_eigen_vector(self, cov_mat):
        """
        Eigenvector and eigen values using numpy. Uses numpy's eigenvalue␣
↪function
        :param cov_mat:
        :return: (eigen_values, eigen_vector)
```

```
        """
        # your code here
        return np.linalg.eig(cov_mat)


    def compute_explained_variance(self, eigen_vals):
        """
        sort eigen values and compute explained variance.
        explained variance informs the amount of information (variance)
        can be attributed to each of  the principal components.
        :param eigen_vals:
        :return: explained variance.
        """
        # your code here
        eigen_val_count = len(eigen_vals)

        eigen_vals = np.sort(eigen_vals)[::-1]
        exp_var = np.empty(eigen_val_count)

        for i in range(eigen_val_count):
            exp_var[i] = (eigen_vals[i]) / np.sum(eigen_vals)

        return exp_var




    def cumulative_sum(self, var_exp):
        """
        return cumulative sum of explained variance.
        :param var_exp: explained variance
        :return: cumulative explained variance
        """
        return np.cumsum(var_exp)

    def compute_weight_matrix(self, eig_pairs, cum_var_exp):
        """
        compute weight matrix of top principal components conditioned on target
        explained variance.
        (Hint : use cumilative explained variance and target_explained_variance␣
↪to find
        top components)

        :param eig_pairs: list of tuples containing eigenvalues and␣
↪eigenvectors,
        sorted by eigenvalues in descending order (the biggest eigenvalue and␣
↪corresponding eigenvectors first).
```

```
        :param cum_var_exp: cumulative expalined variance by features
        :return: weight matrix (the shape of the weight matrix is n X k)
        """
        # your code here
        weight_matrix = np.zeros()
        for i in range(len(eig_pairs)):
            weight_matrix.append(eig_pairs[1])
            exp_vars = eig_pairs(:(i+1))



    def transform_data(self, X_std, matrix_w):
        """
        transform data to subspace using weight matrix
        :param X_std: standardized data
        :param matrix_w: weight matrix
        :return: data in the subspace
        """
        return X_std.dot(matrix_w)

    def fit(self, X):
        """
        entry point to the transform data to k dimensions
        standardize and compute weight matrix to transform data.
        The fit functioin returns the transformed features. k is the number of␣
↪features which cumulative
        explained variance ratio meets the target_explained_variance.
        :param   m X n dimension: train samples
        :return  m X k dimension: subspace data.
        """

        self.feature_size = X.shape[1]

        # your code here

        print(len(matrix_w),len(matrix_w[0]))
        return self.transform_data(X_std=X_std, matrix_w=matrix_w)
```

[**PART A**] Your task involves implementing helper functions to compute *mean, covariance, eigenvector and weights.*

complete `fit()` to using all helper functions to find reduced dimension data.

Run PCA on *fashion mnist dataset* to reduce the dimension of the data.

fashion mnist data consists of samples with *784 dimensions.*

Report the reduced dimension k for target explained variance of **0.99**

[17]:
```
X_train = pickle.load(open('./data/fashionmnist/train_images.pkl','rb'))
y_train = pickle.load(open('./data/fashionmnist/train_image_labels.pkl','rb'))

X_train = X_train[:1500]
y_train = y_train[:1500]
```

[52]:
```
arr = np.array([5,3,5,2,1])
print(arr.sort())
```

```
None
```

[30]:
```
# pca_.compute_mean_vector(X_train)
```

[55]:
```
pca_ = PCA()
cov_matrix = pca_.compute_cov(pca_.standardize(X_train), pca_.
 ↪compute_mean_vector(X_train))
eigenvals, eigen_vectors = pca_.compute_eigen_vector(cov_matrix)
print(pca_.compute_explained_variance(eigenvals))
```

```
[2.17427691e-01 1.43835332e-01 5.74989774e-02 5.19424132e-02
 4.20603475e-02 3.20187752e-02 2.81158094e-02 2.44877442e-02
 1.70584025e-02 1.26649504e-02 1.23017744e-02 1.05555755e-02
 9.49404353e-03 9.35954196e-03 8.51402417e-03 8.04792875e-03
 7.61793024e-03 7.13760021e-03 6.60812584e-03 6.36190607e-03
 6.24877889e-03 5.58230363e-03 5.36012665e-03 5.14833835e-03
 4.89528195e-03 4.70148032e-03 4.39792660e-03 4.27551849e-03
 4.23439543e-03 4.15622433e-03 4.00992459e-03 3.95387074e-03
 3.66560950e-03 3.59799082e-03 3.48912641e-03 3.31610638e-03
 3.28268232e-03 3.19590085e-03 3.07200196e-03 2.96272030e-03
 2.88347949e-03 2.81925768e-03 2.76250080e-03 2.69943208e-03
 2.63559529e-03 2.49945208e-03 2.41119290e-03 2.38936504e-03
 2.36037715e-03 2.27227630e-03 2.22210306e-03 2.20151139e-03
 2.14953188e-03 2.08640702e-03 2.05080686e-03 2.02732718e-03
 1.95152257e-03 1.91188427e-03 1.87961019e-03 1.83335059e-03
 1.79515072e-03 1.76957281e-03 1.74032440e-03 1.72338383e-03
 1.69971303e-03 1.65144049e-03 1.63186533e-03 1.62646279e-03
 1.58331115e-03 1.55483092e-03 1.53404714e-03 1.50958339e-03
 1.48441147e-03 1.43887705e-03 1.41518336e-03 1.38313423e-03
 1.36251340e-03 1.32435084e-03 1.30941761e-03 1.26596693e-03
 1.24249386e-03 1.22025077e-03 1.21314650e-03 1.19345633e-03
 1.17285982e-03 1.14398162e-03 1.12531109e-03 1.11242828e-03
 1.11018973e-03 1.08741263e-03 1.07269191e-03 1.06446780e-03
 1.04961349e-03 1.04005311e-03 1.03609778e-03 1.01735586e-03
 1.00119849e-03 9.64549414e-04 9.58029813e-04 9.42889197e-04
 9.32241719e-04 9.19895921e-04 9.11764693e-04 9.07935668e-04
 9.00748723e-04 8.82087081e-04 8.78958058e-04 8.53840501e-04
 8.51197838e-04 8.39997253e-04 8.28906292e-04 8.24904755e-04
```

8.07688154e-04 8.00798462e-04 7.96436414e-04 7.83131093e-04
7.75171040e-04 7.70748739e-04 7.58815697e-04 7.47555873e-04
7.44536192e-04 7.37002871e-04 7.25860819e-04 7.24222969e-04
7.13680557e-04 7.06731645e-04 7.03402084e-04 6.94063049e-04
6.82245641e-04 6.71147430e-04 6.67731225e-04 6.62096658e-04
6.53837571e-04 6.49062778e-04 6.36486959e-04 6.30768788e-04
6.24265104e-04 6.18978582e-04 6.11072336e-04 6.07733349e-04
6.00496541e-04 5.93031630e-04 5.88467083e-04 5.80663159e-04
5.78024842e-04 5.69594242e-04 5.67862242e-04 5.61633211e-04
5.56047240e-04 5.43013302e-04 5.41444741e-04 5.38317068e-04
5.31172163e-04 5.25186643e-04 5.18510469e-04 5.16860213e-04
5.11748619e-04 5.06620179e-04 5.05720275e-04 4.95219408e-04
4.88129528e-04 4.84803359e-04 4.80475544e-04 4.79275530e-04
4.75586571e-04 4.71154523e-04 4.63902330e-04 4.61166162e-04
4.59882066e-04 4.48114666e-04 4.46474769e-04 4.39375122e-04
4.37394680e-04 4.33916401e-04 4.32117973e-04 4.29725919e-04
4.22902858e-04 4.19353819e-04 4.16919396e-04 4.15657514e-04
4.07646184e-04 4.03171140e-04 3.98326343e-04 3.96435683e-04
3.93814513e-04 3.90474465e-04 3.88879150e-04 3.84670483e-04
3.81289103e-04 3.79324789e-04 3.74220318e-04 3.67303120e-04
3.65735049e-04 3.65071255e-04 3.61990899e-04 3.59869399e-04
3.57588250e-04 3.55852914e-04 3.53060816e-04 3.47111885e-04
3.46325890e-04 3.42430119e-04 3.39274493e-04 3.37320922e-04
3.32890758e-04 3.30348892e-04 3.29120842e-04 3.24015122e-04
3.21861179e-04 3.17973181e-04 3.15003501e-04 3.14146815e-04
3.09184708e-04 3.07612256e-04 3.06468530e-04 3.05257876e-04
3.01751265e-04 2.99625173e-04 2.97282064e-04 2.96126928e-04
2.95143074e-04 2.92905246e-04 2.88625180e-04 2.86384049e-04
2.85508314e-04 2.80469670e-04 2.79838555e-04 2.77943636e-04
2.77160538e-04 2.76930798e-04 2.75011585e-04 2.73421706e-04
2.72417737e-04 2.68886963e-04 2.64400682e-04 2.63026484e-04
2.60883339e-04 2.59160499e-04 2.55896075e-04 2.55715631e-04
2.52394396e-04 2.50903590e-04 2.50163086e-04 2.47780560e-04
2.45385380e-04 2.44387668e-04 2.40895394e-04 2.40412078e-04
2.39457208e-04 2.37636188e-04 2.36404989e-04 2.34324196e-04
2.32558201e-04 2.29456570e-04 2.27958792e-04 2.26978698e-04
2.25955794e-04 2.24630814e-04 2.21040813e-04 2.20614565e-04
2.18859365e-04 2.17116021e-04 2.16673838e-04 2.14549751e-04
2.12366691e-04 2.10519738e-04 2.07690300e-04 2.05576599e-04
2.05334738e-04 2.04188713e-04 2.03656147e-04 2.01993999e-04
2.00947077e-04 1.99230384e-04 1.98385481e-04 1.97284789e-04
1.95205400e-04 1.92472714e-04 1.90542167e-04 1.88869051e-04
1.88570059e-04 1.88192951e-04 1.87676079e-04 1.86382977e-04
1.84496928e-04 1.83262305e-04 1.82293702e-04 1.80531604e-04
1.77828465e-04 1.77093578e-04 1.76337111e-04 1.75664328e-04
1.74062429e-04 1.72745688e-04 1.71466791e-04 1.69929659e-04
1.69475473e-04 1.68372914e-04 1.66575383e-04 1.65293451e-04
1.64615317e-04 1.63362693e-04 1.62946773e-04 1.62188866e-04

1.61274358e-04 1.58699260e-04 1.57927236e-04 1.56859735e-04
1.55913719e-04 1.55093617e-04 1.54016031e-04 1.51224564e-04
1.50732416e-04 1.49861743e-04 1.48848030e-04 1.48586201e-04
1.47048115e-04 1.46705766e-04 1.46318305e-04 1.44608321e-04
1.43433625e-04 1.42952003e-04 1.41644873e-04 1.41340641e-04
1.40430359e-04 1.39986609e-04 1.38365068e-04 1.36625264e-04
1.36275065e-04 1.35375934e-04 1.35107698e-04 1.33889830e-04
1.32871174e-04 1.32121547e-04 1.31169719e-04 1.30060340e-04
1.28305095e-04 1.28063020e-04 1.27480003e-04 1.26186827e-04
1.25352447e-04 1.24302403e-04 1.24002898e-04 1.23270623e-04
1.22612639e-04 1.21053351e-04 1.20826193e-04 1.20172934e-04
1.19366506e-04 1.18730106e-04 1.17705607e-04 1.16794513e-04
1.15216152e-04 1.14988427e-04 1.14512021e-04 1.13989988e-04
1.13415255e-04 1.11859667e-04 1.11324456e-04 1.11129895e-04
1.10687940e-04 1.09856017e-04 1.09099555e-04 1.08583709e-04
1.07306593e-04 1.06567667e-04 1.06119948e-04 1.05478832e-04
1.04503907e-04 1.04038927e-04 1.03031037e-04 1.02498726e-04
1.02244473e-04 1.01514406e-04 1.00325296e-04 9.95470258e-05
9.88902265e-05 9.86904637e-05 9.82749433e-05 9.75273032e-05
9.66620537e-05 9.52974195e-05 9.49241323e-05 9.45627775e-05
9.36809133e-05 9.25530236e-05 9.23814108e-05 9.16340087e-05
9.07601351e-05 9.03209352e-05 8.97127386e-05 8.95270506e-05
8.84999451e-05 8.82485135e-05 8.77963128e-05 8.69867707e-05
8.63337116e-05 8.54083400e-05 8.49881093e-05 8.44494913e-05
8.39753042e-05 8.31868050e-05 8.28331965e-05 8.23219219e-05
8.20664534e-05 8.13334644e-05 8.08523898e-05 8.06621009e-05
7.96474844e-05 7.93219058e-05 7.88313263e-05 7.80323504e-05
7.79264132e-05 7.75526753e-05 7.66669853e-05 7.65708459e-05
7.64975607e-05 7.57043126e-05 7.50840525e-05 7.46942438e-05
7.40950529e-05 7.32631665e-05 7.31652972e-05 7.22992573e-05
7.18712423e-05 7.16388305e-05 7.10653499e-05 7.08858140e-05
6.98424895e-05 6.95888992e-05 6.90161422e-05 6.80650470e-05
6.74236410e-05 6.72415328e-05 6.70311675e-05 6.66722205e-05
6.61355148e-05 6.54172418e-05 6.49368703e-05 6.43132210e-05
6.36509792e-05 6.35110347e-05 6.34987255e-05 6.31227982e-05
6.21920862e-05 6.20123229e-05 6.18260940e-05 6.12907336e-05
6.07636861e-05 6.00048504e-05 5.99184898e-05 5.98309146e-05
5.91847424e-05 5.87544522e-05 5.85536858e-05 5.83679783e-05
5.80061607e-05 5.75514849e-05 5.69218392e-05 5.67029335e-05
5.58120723e-05 5.55348381e-05 5.53016344e-05 5.49634361e-05
5.42339082e-05 5.41405671e-05 5.35078347e-05 5.31762105e-05
5.28922667e-05 5.25827585e-05 5.23930500e-05 5.17905759e-05
5.16393213e-05 5.11315183e-05 5.08054202e-05 5.04437061e-05
5.01918821e-05 5.00304755e-05 4.95618300e-05 4.91072429e-05
4.85742230e-05 4.84836114e-05 4.81360187e-05 4.79125286e-05
4.76205013e-05 4.73517652e-05 4.71649072e-05 4.63301091e-05
4.60251231e-05 4.57334239e-05 4.54488218e-05 4.52545673e-05
4.50607462e-05 4.46475076e-05 4.42447975e-05 4.41611961e-05

4.38511325e-05 4.37065037e-05 4.35421267e-05 4.30151830e-05
4.29408449e-05 4.17400629e-05 4.16237693e-05 4.11787824e-05
4.09283339e-05 4.07675150e-05 4.06300421e-05 4.01399872e-05
3.97951467e-05 3.96611234e-05 3.91536072e-05 3.91376549e-05
3.87947383e-05 3.86550884e-05 3.83860207e-05 3.82271548e-05
3.80444083e-05 3.75320995e-05 3.71890826e-05 3.70196880e-05
3.66612550e-05 3.63669801e-05 3.60013437e-05 3.59614496e-05
3.57141927e-05 3.53992537e-05 3.49747705e-05 3.46898774e-05
3.44754195e-05 3.41786773e-05 3.40310937e-05 3.37944663e-05
3.34813180e-05 3.33465088e-05 3.30971498e-05 3.28765398e-05
3.27530246e-05 3.22233056e-05 3.19877821e-05 3.19157078e-05
3.16130519e-05 3.14597114e-05 3.11670810e-05 3.07924411e-05
3.04706675e-05 3.02199441e-05 2.99590714e-05 2.97862573e-05
2.95397501e-05 2.93482467e-05 2.90710893e-05 2.88653140e-05
2.87644037e-05 2.84780593e-05 2.83374140e-05 2.81769727e-05
2.79216152e-05 2.76048207e-05 2.73906138e-05 2.70584178e-05
2.68033803e-05 2.67148995e-05 2.66937481e-05 2.64181277e-05
2.61458755e-05 2.58548827e-05 2.57824351e-05 2.55664944e-05
2.53817743e-05 2.52410257e-05 2.51161611e-05 2.49178416e-05
2.46131020e-05 2.43481736e-05 2.41592341e-05 2.41269490e-05
2.38645417e-05 2.36660547e-05 2.34899795e-05 2.33796784e-05
2.32145917e-05 2.28952326e-05 2.26968732e-05 2.26584286e-05
2.21772668e-05 2.20937103e-05 2.19150217e-05 2.18518288e-05
2.16950181e-05 2.14025906e-05 2.12567553e-05 2.10610542e-05
2.08800920e-05 2.06406322e-05 2.04629001e-05 2.01844744e-05
2.00677466e-05 2.00369201e-05 1.98316971e-05 1.95341669e-05
1.94941377e-05 1.93516500e-05 1.92395408e-05 1.90450484e-05
1.89518494e-05 1.87947448e-05 1.87110461e-05 1.85862765e-05
1.83490022e-05 1.81723763e-05 1.80406774e-05 1.79938804e-05
1.78404849e-05 1.75465158e-05 1.74071885e-05 1.72322528e-05
1.69801758e-05 1.69423365e-05 1.68303778e-05 1.66393758e-05
1.65566857e-05 1.63270408e-05 1.62459899e-05 1.61530585e-05
1.59955508e-05 1.58284467e-05 1.56497710e-05 1.55441994e-05
1.53705622e-05 1.52413380e-05 1.51167772e-05 1.49817568e-05
1.48031017e-05 1.46779004e-05 1.45343314e-05 1.44787589e-05
1.43686264e-05 1.42556906e-05 1.40452072e-05 1.39160977e-05
1.37828412e-05 1.36944564e-05 1.35373304e-05 1.34098465e-05
1.33138768e-05 1.32687475e-05 1.31320034e-05 1.30234671e-05
1.29325626e-05 1.27456743e-05 1.26821733e-05 1.25422115e-05
1.24047029e-05 1.23899707e-05 1.22667177e-05 1.21308574e-05
1.19333976e-05 1.18589508e-05 1.16742764e-05 1.14859361e-05
1.14527733e-05 1.12522625e-05 1.10959318e-05 1.10357916e-05
1.09651587e-05 1.09066195e-05 1.07790671e-05 1.06443568e-05
1.05390644e-05 1.04156338e-05 1.03562871e-05 1.01990508e-05
1.01641336e-05 1.00299442e-05 9.88427956e-06 9.72232068e-06
9.62027584e-06 9.55855997e-06 9.44007079e-06 9.33651074e-06
9.27972214e-06 9.22093666e-06 9.09607790e-06 8.84623673e-06
8.78622340e-06 8.72831403e-06 8.59454126e-06 8.50276876e-06

```
8.43813259e-06 8.36227671e-06 8.23682263e-06 8.17178071e-06
8.10218215e-06 7.98906102e-06 7.94637549e-06 7.85569201e-06
7.70868979e-06 7.54135380e-06 7.50962289e-06 7.42483844e-06
7.32263293e-06 7.20992183e-06 7.15646376e-06 7.10480945e-06
7.06861664e-06 6.92928776e-06 6.90594291e-06 6.71741101e-06
6.61449707e-06 6.53067751e-06 6.50804761e-06 6.43471356e-06
6.37125466e-06 6.21717802e-06 6.14948167e-06 6.08965311e-06
6.05481342e-06 5.95647833e-06 5.82748402e-06 5.75920625e-06
5.68337243e-06 5.58908706e-06 5.51216411e-06 5.45257288e-06
5.36669795e-06 5.31061457e-06 5.19787336e-06 5.18161892e-06
5.10206647e-06 4.95985934e-06 4.94419981e-06 4.87100476e-06
4.76999679e-06 4.73331622e-06 4.67651480e-06 4.58787148e-06
4.56806410e-06 4.49197282e-06 4.45837621e-06 4.33016575e-06
4.24556544e-06 4.15559822e-06 4.14584539e-06 4.07508055e-06
4.04281354e-06 3.98205070e-06 3.87708212e-06 3.81400239e-06
3.78782165e-06 3.67201286e-06 3.62831176e-06 3.48971806e-06
3.38532385e-06 3.32800523e-06 3.24977511e-06 3.19316135e-06
3.05174398e-06 2.97948585e-06 2.94484513e-06 2.90992506e-06
2.83437104e-06 2.75679459e-06 2.71199579e-06 2.62881401e-06
2.56404042e-06 2.52245406e-06 2.47220209e-06 2.37983152e-06
2.32504759e-06 2.29359638e-06 2.11113250e-06 2.08500956e-06
1.97857981e-06 1.86532445e-06 1.86008452e-06 1.79109294e-06
1.67460296e-06 1.65464538e-06 1.42228895e-06 1.27040416e-06
8.24686692e-07 3.08802783e-07 2.08782082e-07 1.36347176e-07]
```

[6]:
```
pca_handler = PCA(target_explained_variance=0.99)
X_train_updated = pca_handler.fit(X_train)
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-6-678395cd90d4> in <module>
  1 pca_handler = PCA(target_explained_variance=0.99)
----> 2 X_train_updated = pca_handler.fit(X_train)


NameError: name 'X_train' is not defined
```

[ ]:
```

```