8. Dimensionality Reduction Using PCA

PCA finds (orthogonal) directions of maximal variation in the data. In this problem we're going to project our data onto the principal components and explore the effects on performance.

You will be working in the files `part1/main.py` and `part1/features.py` in this problem

Project onto Principal Components

3/3 points (graded)
Fill in function `project_onto_PC` in **features.py** that implements PCA dimensionality reduction of dataset X.

Note that to project a given $n \times d$ dataset X into its k-dimensional PCA representation, one can use matrix multiplication, after first centering X:

$$\widetilde{X}V$$

where $\widetilde{X}$ is the centered version of the original data X and V is the $d \times k$ matrix whose columns are the top k eigenvectors of $\widetilde{X}^T\widetilde{X}$. This is because the eigenvectors are of unit-norm, so there is no need to divide by their length.

You are given the full principle component matrix V as `pcs` in this function.

Available Functions: You have access to the NumPy python library as `np` and the function `center_data` which returns a centered version of the data, where each feature now has mean = 0

```
def project_onto_PC(X, pcs, n_components):
    """
    Given principal component vectors pcs = principal_components(X)
    this function returns a new data array in which each sample in X
    has been projected onto the first n_components principcal components.
```

```
    """
    # TODO: first center data using the centerData() function.
    # TODO: Return the projection of the centered dataset
    #       on the first n_components principal components.
    #       This should be an array with dimensions: n x n_components.
    # Hint: these principal components = first n_components columns
    #       of the eigenvectors returned by principal_components().
    #       Note that each eigenvector is already be a unit-vector,
    #       so the projection may be done using matrix multiplication.
    return center_data(X)@pcs[:,:n_components]
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Test results

Hide output

CORRECT

Test: random 1 components

Testing random 10x10 matrix, 1 dimension of principal components

Output:

```
[[-0.52419822]
 [ 0.53481097]
 [ 0.54061591]
 [-0.55761143]
 [-0.29533271]
 [ 0.55599834]
 [-0.83759467]
 [-0.05948855]
 [ 0.46507569]
 [ 0.17772467]]
Test completed
```

Test: random 2 components

Testing random 10x10 matrix, 2 dimensions of principal components

Output:

```
[[ 0.54441136 -0.17290404]
 [-0.34595884  0.27106476]
 [ 0.54264276  0.25105358]
 [-0.74152861 -0.27102511]
 [-0.66497236  0.80528085]
 [-0.30316319 -0.56314011]
 [-0.47544417 -0.48207973]
 [ 0.59194911  0.21532154]
 [ 0.56845961 -0.44232695]
 [ 0.28360434  0.3887552 ]]
Test completed
```

Test: random 3 components pos and neg

Testing random 10x10 matrix, n dimensions of principal components

Output:

```
[[ 7.13198077+0.j -3.04033825+0.j  2.40645114+0.j]
 [-5.40042329+0.j  2.16724206+0.j  5.19625907+0.j]
 [-3.29184546+0.j -5.9819278 +0.j -2.03819762+0.j]
 [-2.70305158+0.j  2.22916257+0.j -4.00683937+0.j]
 [ 4.26333956+0.j  4.62586142+0.j -1.55767322+0.j]]
Test completed
```

Test: random n components

Testing random 10x10 matrix, n dimensions of principal components

Output:

```
[[-3.46874904e-01  1.91029330e-01 -4.75621960e-02  4.89326069e-01
   2.20386309e-01 -4.92054146e-02 -1.68651663e-01 -8.57048088e-02
   9.09446118e-04  3.30960787e-16]
 [-5.92738843e-01 -4.62759945e-01  1.88237658e-01  6.26206691e-02
  -4.65572234e-01  5.84788206e-02  7.65262786e-02 -1.06835282e-01
  -3.30720557e-02  8.37171450e-18]
 [-5.29517119e-03  1.06222940e-01 -3.59493510e-01  1.71250941e-01
  -3.95248504e-01 -2.71364054e-01 -7.40393722e-03  1.21765245e-01
   8.05757100e-03  3.77769957e-17]
 [ 4.36083566e-01 -5.19320041e-01 -3.95690217e-01 -1.34367479e-01
  -1.27756086e-01  2.25950483e-01 -8.26885060e-02  1.22008783e-02
   8.04785294e-02  6.65412926e-17]
 [-8.00522040e-01 -3.84580661e-01 -7.06637487e-02 -4.47870829e-02
   4.66783341e-01 -1.47491506e-02  9.31054942e-02  7.40027905e-02
   3.57992460e-02 -2.75605405e-16]
 [ 4.89903221e-01  6.37021303e-02  5.53824436e-01 -2.35205223e-01
   6.95804744e-02 -2.18983909e-01  3.25439950e-02 -8.07836511e-02
   7.56421936e-02 -1.62907737e-16]
 [ 4.02954411e-01 -4.08884548e-01 -2.17087188e-01 -3.65804996e-01
   2.04103064e-01 -9.35235446e-02 -4.43927081e-02 -1.44012102e-02
  -1.12653555e-01 -2.30819387e-17]
 [-9.01735320e-02  9.51994305e-01 -5.48130208e-01 -2.38663040e-01
   1.42907688e-02  8.99158280e-02  7.52517087e-02 -5.93129761e-02
  -2.19272357e-04 -2.19136041e-16]
 [-2.51201287e-01  4.16281307e-01  6.97264330e-01 -2.45339457e-01
  -1.01628583e-01  1.47036517e-01 -8.58797331e-02  1.02920911e-01
  -2.40554697e-02 -3.61194070e-17]
 [ 7.57864580e-01  4.63151821e-02  1.99300642e-01  5.40969599e-01
   1.15061449e-01  1.26444424e-01  1.11589071e-01  3.61481045e-02
  -3.08866331e-02 -2.71414859e-16]]
Test completed
```

Hide output

Submit You have used 1 of 20 attempts

✔ Correct (3/3 points)

Note: we only use the training dataset to determine the principal components. It is **improper** to use the test dataset for anything except evaluating the accuracy of our predictive model. If the test data is used for other purposes such as selecting good features, it is possible to overfit the test set and obtain overconfident estimates of a model's performance.

Testing PCA

1/1 point (graded)
Use `project_onto_PC` to compute a 18-dimensional PCA representation of the MNIST training and test datasets, as illustrated in `main.py`.

Retrain your softmax regression model (using the original labels) on the MNIST training dataset and report its error on the test data, this time using these 18-dimensional PCA-representations rather than the raw pixel values.

If your PCA implementation is correct, the model should perform nearly as well when only given 18 numbers encoding each image as compared to the 784 in the original data (error on the test set using PCA features should be around 0.15). This is because PCA ensures these 18 feature values capture the maximal amount of variation from the original 784-dimensional data.

Error rate for 18-dimensional PCA features = 0.1483 ✔

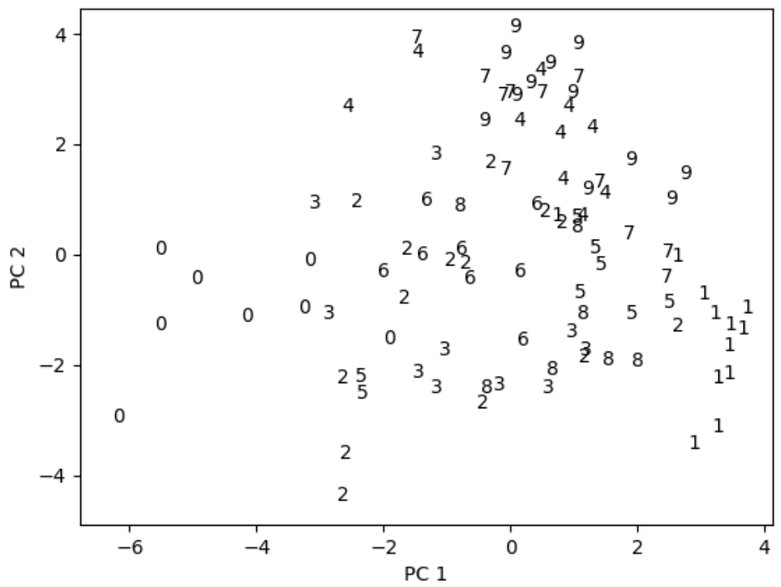
Submit You have used 1 of 2 attempts

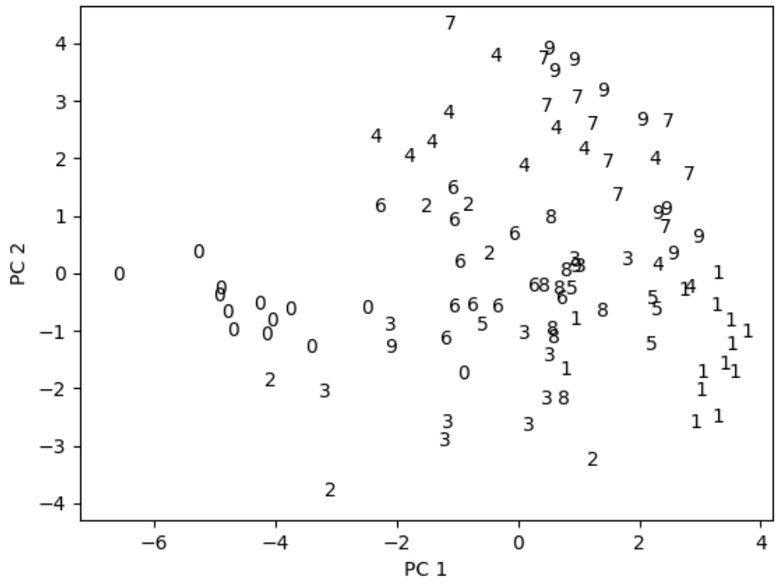
✔ Correct (1/1 point)

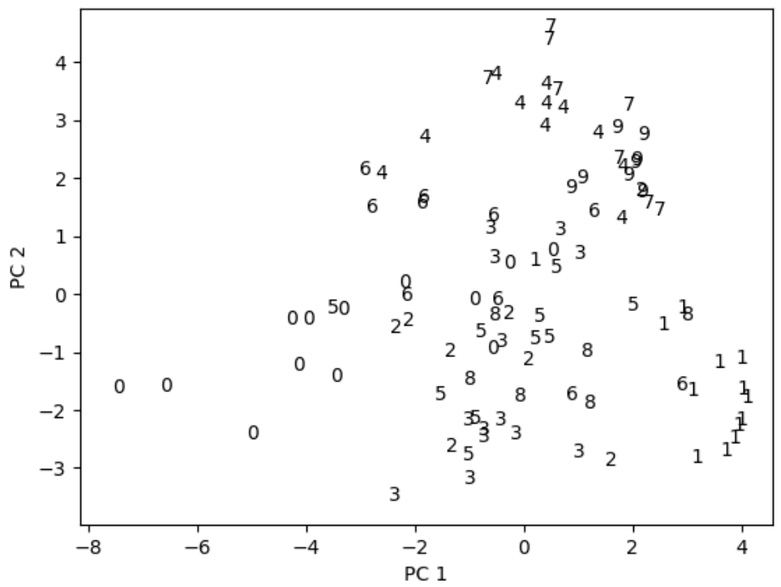
Testing PCA

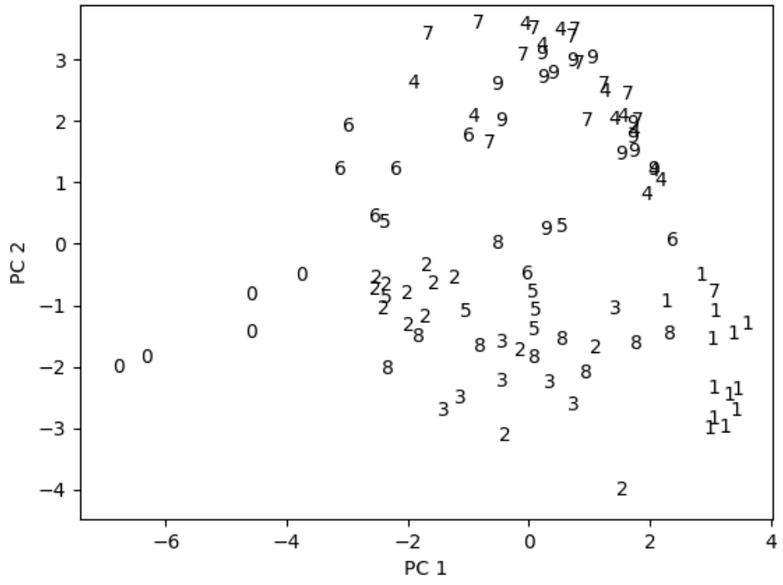
1/1 point (graded)
Use `plot_PC` in **main.py** to visualize the first 100 MNIST images, as represented in the space spanned by the first 2 principal components of the training data.

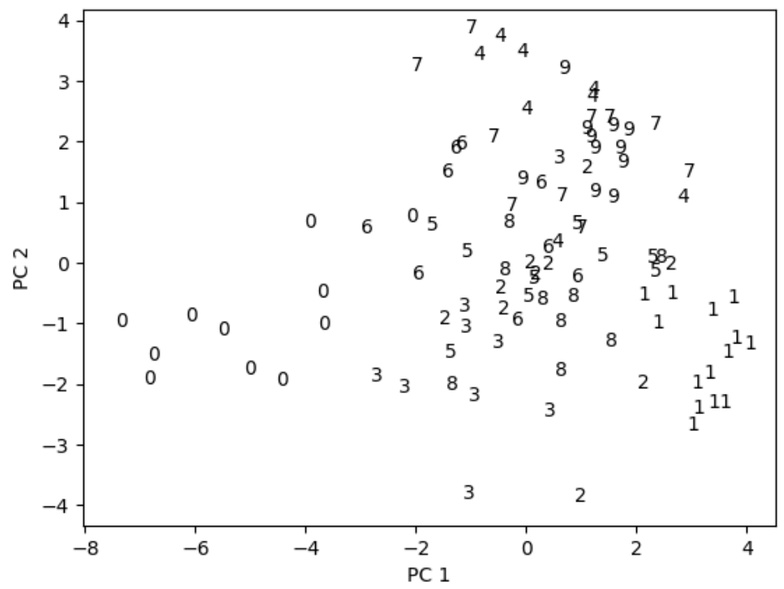
What does your PCA look like?


PC 2
PC 1


PC 2
PC 1


PC 2
PC 1


PC 2
PC 1



Use the calls to `plot_images()` and `reconstruct_PC` in **main.py** to plot the reconstructions of the first two MNIST images (from their 18-dimensional PCA-representations) alongside the originals.

Submit You have used 1 of 2 attempts

✔ Correct (1/1 point)

Remark: Two dimensional PCA plots offer a nice way to visualize some global structure in high-dimensional data, although approaches based on nonlinear dimension reduction may be more insightful in certain cases. Notice that for our data, the first 2 principal components are insufficent for fully separating the different classes of MNIST digits.

Discussion

Hide Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Project 2: Digit recognition (Part 1) / 8. Dimensionality Reduction Using PCA

Add a Post

Show all posts ▼ by recent activity ▼

Testing PCA
Can you check if you have the right answer? I tried doing this using both settings: first updating the labels to mod3 before running softmax regresssion and taking mod3 after ... 7

test.py's requirement for PCA is wrong
I guess the course staff has entered the wrong test result, correct code for `test.py` line 237-241 should be exp_res = np.array([[2, 0, 0], [0, 0, 0], [-2, 0, 0]]) 4

? Project onto Principal Components: principle component typo 2

Nonlinear dimension reduction
Here some examples of nonlinear dimension redution: isomap: https://web.mit.edu/cocosci/Papers/sci_reprint.pdf t-sne: https://en.wikipedia.org/wiki/T-distributed_stochasti... 1
Community TA

Learn About Verified Certificates