Connor McCurley

EEE 6814 **Homework 5** Fall 2019

This assignment tasked us with applying multi-resolution PCA as a pre-processing procedure for Fashion-MNIST classification. To be honest, I never fully understood the implementation of PCA-M demonstrated in the paper (the steps were really not clear). So I implemented my own version of multi-resolution PCA (which may or may not be valid). To do this, I took the full-resolution images (28x28) and generated (14x14), (7x7) and (3x3) lower-resolution versions. For each image resolution, I estimated the means and covariances of the training set. The full set of eigenvectors of the covariance matrix are commonly referred to as the "eigenface space" in the literature. Projecting individual images onto the full space provides weights which represent the amount each eigenvector contributes toward reconstructing the image. To generate features for an image, the following was employed:

1. Vectorize the (14x14) image
2. Take the first three eigenvectors multiplied by the images' reconstruction weight
3. Concatenate those four images
4. Repeat for the (7x7) and (3x3) images
5. Concatenate all features to make a 1016 length feature vector for the image

This process is exhibited in Figures **??**,**??**, and **??**. The images on the left show the full (28x28) sample. In the right images, the top row (from left to right) corresponds to the full (14x14) resolution image, the reconstruction contribution of the first principal vector, followed by the second and third. The middle row corresponds to the same for the (7x7) image and the bottom row demonstrates the (3x3) resolution image. Each of the images on the right are vectorized and concatenated to form 1016 length feature vectors.
Feature were generated as demonstrated above. Additionally, I tested taking the 2D FFT of each component image before training the network. Again, this results in 1016 dimensional feature vectors for each image. It should be discussed that while the dimensionality was actually increased (from 784) in this scenario, it was believed that the features would be more discriminative than the raw images alone. A single network architecture was implemented as described in project 1, consisting of layers (input-256-128-100-10). ReLU activation functions were used in all layers of the network, excluding the output. Cross-entropy was the implemented cost function and the Adamax optimizer (with and initial learning rate of $\eta = 0.01$) implementation in Pytorch was used to update the weights. Early stopping was applied when the validation error began to increase. The training set consisted of 60000 samples, the validation, 9000 samples and the hold-out test set, 10000 samples. The number of samples for each class were evenly distributed in each data split. Ten models were trained for both feature scenarios and the best model (on validation) was applied to the test set. Results are shown in the form of confusion matrices in Figure **??** and compared to results obtained from project 1 in Figure **??**.
From the figures, it can be observed that the base model exhibited loss of 0.61, the base PCA retaining 100 dimensions showed 0.34, PCA-M demonstrated 0.38, and the PCA-M with frequency domain features exhibited 0.88 cross-entropy loss. It is still my belief the PCA-100 model trained for project 1 was "lucky". I find it difficult to believe that this model could achieve a lower error than some of the other methods compared. PCA-M, however, demonstrated statistically equivalent results. While not shown, the network trained with PCA-M features consistently reached the same performance for all 10 training sessions. Additionally, while the dimensionality of the features actually increased, it appears that the PCA-M features were actually better descriptors for discrimination than the raw samples.

Again, PCA does not conserve discriminability through its transformation of data. Therefore, dimensionality reduction methods which enforce class separability could be investigated as a preprocessing procedure in future work.