Homework 1

Due February 7 at 11 pm

1. (Centering) To analyze what happens if we apply PCA without centering, let $\tilde{x}$ be a d-dimensional vector with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma_{\tilde{x}}$ equal to the identity matrix. If we compute the eigendecomposition of the matrix $\mathrm{E}(\tilde{x}\tilde{x}^T)$ what is the value of the largest eigenvalue? What is the direction of the corresponding eigenvector?

2. (Low-rank covariance matrix) Let $\tilde{x}$ be a d-dimensional vector, show that if its covariance matrix $\Sigma_{\tilde{x}}$ is rank $r < d$, then $\tilde{x}$ is restricted to a hyperplane of dimension r, in the sense that it belongs to the hyperplane with probability one. Recall that by Chebyshev's inequality, for any positive constant $c > 0$ and any random variable $\tilde{a}$ with bounded variance,

$$\mathrm{P}\left(\left(\tilde{a} - \mathrm{E}\left(\tilde{a}\right)\right)^2 \geq c\right) \leq \frac{\mathrm{Var}\left(\tilde{a}\right)}{c}. \tag{1}$$

3. (Dimensionality reduction) Data containing a large number of features can be difficult to analyze and process. The goal of dimensionality-reduction techniques is to embed the data points in a low-dimensional space where they can be described with a small number of variables. Here we study *linear* dimensionality reduction, where the lower-dimensional representation is obtained by computing the inner products of each data point with a small number of basis vectors. We will show that PCA provides an optimal choice for the linear transformation.

   (a) Our proof will be based in the following linear-algebraic result. Let $A \in \mathbb{R}^{d \times d}$ be a symmetric matrix, and $u_1, \ldots, u_k$ be its first k eigenvectors. For any set of k orthonormal vectors $v_1, \ldots, v_k$

   $$\sum_{i=1}^{k} u_i^T A u_i \geq \sum_{i=1}^{k} v_i^T A v_i. \tag{2}$$

   Show that this follows from the spectral theorem using induction.

   (b) Let us consider an arbitrary dataset $X = \{x_1, \ldots, x_n\}$ of d-dimensional points. For a fixed dimension $k < d$, and any set of k orthonormal vectors $v_1, \ldots, v_k$ show that the sum of the sample variances of the k first principal components,

   $$\sum_{i=1}^{k} \mathrm{var}(\mathrm{pc}[i]) \geq \sum_{i=1}^{k} \sigma^2_{X_{v_i}}, \tag{3}$$

   where $\sigma^2_{X_{v_i}}$ is the sample variance of $\left\{v_i^T x_1, \ldots, v_i^T x_n\right\}$. This establishes that projecting onto the principal directions is the linear transformation that preserves the most variance in the data.

4. (Nearest neighbors in lower dimensions) To illustrate a possible use of PCA-based dimensionality reduction, we consider the problem of face classification. The nearest-neighbor algorithm is a classical method to perform classification. When applied on d-dimensional points, it requires computing distances in d dimensions, which can. be very computationally expensive if d is large. Here you will use PCA to perform nearest neighbors in a lower-dimensional space. You will find the relevant code in the folder `faces` .

   (a) Complete the `compute_nearest_neighbors()` function in `nearest_neighbors.py` that finds the image in the training data that is closest to a given test image. Include the generated images in your submitted homework. In this question, do not apply dimensionality reduction.

   Create a new file in which to write the code for the remaining questions. Include all plots in your submitted homework.

   (b) Generate a plot of k vs. σ_k^2, where σ_k^2 is the variance of the kth principal component of the data (e.g., σ_1^2 is the largest variance). You can limit the x axis to a reasonable number.

   (c) Plot (using `plot_image_grid()` in `plot_tools.py`) the vectors corresponding to the top 10 principal directions of the data. Your principal direction vectors should be elements of $\mathbb{R}^{4096}$ (i.e., they should represent images).

   (d) Use the variance of principal directions plot to determine a relatively small number k of principal components that explains the training data reasonably well. Project the training data and the test data onto the first k principal components, and run nearest neighbors for each test image in this lower dimensional space. Include your choice for k, and the plots of your nearest neighbor results in your submitted homework document. You should use the code from `nearest_neighbors.py` to generate your image plots.

   (e) Let's explicitly study how accuracy changes with the number of principal components k used to project before computing nearest neighbhor. We will divide our dataset of 400 elements into 350 for training and 50 for test and compute the classification accuracy of these 50 with different values of k. The indices of 50 test elements and the minimum set of k values are specified as comments at the end of `nearest_neighbors.py`. Please feel free to include more k values. Generate a plot of accuracy vs k

   (f) Repeat the analysis above, but now with a train and test dataset that is corrupted by Gaussian noise with standard deviation 150/255. Sample code for corrupting the data is provided in `nearest_neighbors.py`. Do you observe a similar pattern as in the previous part? Can you explain what you are observing?

   Some notes to keep in mind:

   (a) The function `np.linalg.eig` might return complex eigenvectors.

   (b) The data points in the training and test data are given as rows.

   (c) Include all new code (or functions) you have filled in your final PDF.