Assignment 3, CSCI 471/571 Fall 2020

Your name here

Due: October 23, 11:59 p.m.

Group work following the guidelines in the Syllabus is okay. List your collaborators.
Total points: 40

The following sections should be computed by hand. Show your work.

Principal components analysis

1. (Trace basics) The trace of a square $n \times n$ matrix A is equal to the sum of the diagonal entries. In math,

$$\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{ii}.$$

It is a fact that $\mathrm{Tr}(A) = \sum_{i=1}^{n} \lambda_i$, where $(\lambda_i)_{i=1}^{n}$ are the eigenvalues of A. Show that:

1. *[1 points]* $\mathrm{Tr}(A + B) = \mathrm{Tr}(A) + \mathrm{Tr}(B)$.
2. *[3 points]* If $X = USV^T$ is the SVD of the $n \times d$ matrix X, with singular values $s_1 \geq s_2 \geq \ldots \geq s_{\min\{n,d\}} \geq 0$, show that $\mathrm{Tr}(X^T X) = \sum_i s_i^2$.

2. (Frobenius norm) The Frobenius norm is a matrix norm that shows up in PCA and other unsupervised methods, among other star appearances. For an $m \times n$ matrix A, it is defined as

$$\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}.$$

1. *[2 points]* Show that $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij}^2}$.
2. *[1 points]* Explain why the Frobenius norm is the same as viewing the matrix as a vector and using one of the usual vector norms.
3. *[2 points]* Show that if U is an $m \times m$ orthogonal matrix and $\vec{v}$ is an m-vector, then $\|U\vec{v}\|_2 = \|\vec{v}\|_2$. Here, $\|\vec{v}\|_2 = \sqrt{\sum_{i=1}^{m} v_i^2}$ is the usual Euclidean norm. You can think of $U\vec{v}$ as a rotated version of the vector. (Hint: use the inner-product formula for $\|\cdot\|_2$.)
4. *[2 points]* Show that if U is an $m \times m$ orthogonal matrix, then $\|UA\|_F = \|A\|_F$. Again, applying U results in a rotated version of A (where the columns are all rotated).

3. (Variance and PCA) This question shows how the squared singular values may be thought of as variances. Let $w_i \in \mathbb{R}$ for $i = 1, \ldots, n$ be iid samples of a scalar random variable W. Recall that we defined

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (w_i - \hat{\mu})^2 \tag{1}$$

as an unbiased estimator of the variance $\mathrm{Var}[W]$, where $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} w_i$ is an unbiased estimator of its mean $\mathbb{E}[W]$. Given an $n \times d$ data matrix X, PCA is the method of taking the SVD of a matrix $\bar{X} = USV^T$, where $\bar{X}$ is equal to X minus its column averages.

1. *[2 points]* Define $\hat{\sigma}_j$ as the sample variance of the jth feature, for $j = 1, \ldots, d$. You are given n samples of each feature: these are stored in the jth column of X. Show that the sum of the variance across all features

$$(n-1)\sum_{j=1}^{d} \hat{\sigma}_j = \|\bar{X}\|_F^2.$$

2. *[1 points]* Show that the total variance is also equal to the sum of the squared singular values

$$(n-1)\sum_{j=1}^{d} \hat{\sigma}_j = \sum_i s_i^2,$$

where s_i are the singular values. (Hint: use one of the formulas from an earlier problem.)

Programming exercises

For the following, you should program in Python and may use the `numpy` package and `matplotlib` for plotting, and `pandas` for loading data, but *you may not use any of the built-in least-squares solvers or anything from* `sklearn`. Any output from your program (displays of matrices or vectors and plots) must be included in your pdf submission. Your code must be submitted as described in the Syllabus. All plots should be legible with axes labeled and legends if there are multiple things plotted.

4. (Neuroscience example) This dataset consists of the average firing rates of n neurons measured at d contiguous points in time. These neurons are in the motor cortex of a macaque that is reaching out towards a target and bringing its arm back. Motor cortex activity changes during movements in ways that is thought to "represent" the movement in the brain. In the paper where these data were described, the authors developed a more sophisticated technique called jPCA and applied it to the data.

We will apply standard PCA to the dataset provided. There is starter code to load the X matrix. Applying PCA to this example will result in left singular vectors (neuron principal components) that capture covariation among neurons and right singular vectors (time principal compoents) that capture covariation among points in time.

1. *[2 points]* Write a function that returns the PCA of the data matrix. See the starter code. Follow the function specifications precisely. Note that in the `numpy` documentation of `np.linalg.svd`, V^H is another way of writing V^T (it refers to the conjugate or Hermitian transpose, but since our data are real not complex numbers, we don't have to worry about this). Use the `full_matrices=False` option in `np.linalg.svd` to compute the so-called economy SVD.

2. *[1 points]* Use the `np.allclose` command to ensure that PCA correctly reconstructs the matrix. In other words, check that $X = USV^T + \vec{1}\vec{\mu}^T$ when U, S, V are the matrices output by your function, $\vec{\mu}$ are the column means stored in a d-vector, and $\vec{1}$ is the n-vector of ones.

3. *[3 points]* Write a function that returns the cumulative fraction of variance explained by components. This will return a vector, where entry i is the fraction of variance explained by components $\{1, \ldots, i\}$ but leaving out components $\{i+1, \ldots, \text{rank}(X)\}$.

4. *[2 points]* Make a plot fraction of variance explained (y-axis) versus number of components included (x-axis). Plot a horizontal line at 95%. How many components do you need to capture $\geq$95% of the variance?

5. *[4 points]* Project the data onto the first two neuron principal components. Construct a new $2 \times d$ matrix of data

$$Y = \begin{bmatrix} \vec{u}_1^T \bar{X} \\ \vec{u}_2^T \bar{X} \end{bmatrix},$$

where $\vec{u}_i$ is the ith left singular vector. Each of the rows of this matrix can be seen as timeseries of components 1 and 2 rather than the n neurons.

Plot of the matrix Y two different ways. First, plot the entries (y-axis) verus time (x-axis) using `matplotlib.pyplot.plot`.

Second, plot the *population trajectory*, i.e. plot the columns (loadings onto component 1 and 2) for all time points on the same axes. Axes should be labeled by component identity. Use the `matplotlib.pyplot.scatter` and color the points by time (1 through 61). Add a colorbar.

6. *[2 points]* (Interpretation) Interpret the plot of the trajectory in terms of the movement the monkey made. Do the neurons make a movement as well?

5. This problem has you apply ridge regression on a real-world dataset. This dataset is the one called `prostate` in the ESL book.

1. *[1 points]* Estimate the means and standard deviations of the features using just the training data. Print out your answers. These should both be length d vectors.

2. *[2 points]* Standardize the training and testing data matrices `X_train, X_test` using the parameters you calculated in step 1. **Do not recalculate them with the testing data!** Is the testing data standardized "as well" as the training data?

3. *[3 points]* Write a function that takes in the standardized data along with a ridge parameter α (we aren't calling it `lambda` because that's a reserved word in python) that must be positive. Your function should return a vector $\vec{\beta} = [\beta_0, \beta_1, \ldots, \beta_d]^T$ of length $d+1$. When X is augmented to include a column of 1s (thus allowing us to fit the intercept term β_0), the normal equations change to be

$$(X^T X + \alpha D)\vec{\beta} = X^T \vec{y},$$

where $D = \text{diag}(0, 1, \ldots, 1)$ is a $(d+1) \times (d+1)$ diagonal matrix that ensures the ridge penalty misses β_0. Your function should solve this equation to find β_{ridge}.

4. *[3 points]* Use your function with $\alpha = 1$ to fit an estimator to the standardized data. Report the mean squared error (MSE) over the training and testing sets. Now, change to $\alpha = 0.1$ and $\alpha = 10$, and report those errors as well.

5. *[3 points]* Discuss which values of α resulted in best training and testing errors. If you were to use this approach to select your α, do you think that the error on the testing set provides a good estimate of the performance on data you truly haven't seen? Why or why not.