CSCI4390-6390 Assign3

Assign3: Non-Linear Dimensionality Reduction and Linear Regression

Due Date: Oct 9, before midnight (11:59:59PM, Alofi Time; GMT-11)

Both Part I and II have to be done by all sections. Differences have been specified by **CSCI4390** and **CSCI6390** labels.

You will use the Appliances energy prediction data set. You should ignore the first attribute, which is a date-time variable, and you should also remove the last attribute, which is a duplicate of the previous one. For Kernel PCA, you will use all remaining attributes. For Linear Regression, use the first attribute (after removing the date-time variable), which denotes the Appliances Energy Use, as the response variables, with the remaining attributes as predictor variables.

Part I: Kernel Principal Components Analysis (50 points)

You will implement the Kernel PCA (KPCA) algorithm as described in Algorithm 7.2 (Chapter 7, page 208). You need to compute the kernel and then center it, followed by extracting the dominant eigenvectors, which will give you the components of the directions in feature space. Next you will project and visualize the data. To compute the principal components (PCs) from the kernel matrix, you may use the inbuilt numpy function eigh.

Using the linear kernel (i.e., polynomial kernel with degree $q = 1$ and $c = 0$; in other words $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$), how many dimensions are required to capture $\alpha = 0.95$ fraction of the total variance? For the same linear kernel, compute the projected points along the first two kernel PCs, and create a scatter plot of the projected points.

Next, use the covariance matrix for the original data to compute the regular principal components, i.e., for the covariance matrix. Project the data onto the first two PCs. How do the eigenvalues and the projection compare with that obtained via Kernel PCA with linear kernel?

Finally, use the gaussian kernel and repeat the exercise for kernel PCA. Project the points onto the first two PCs and plot the scatter plot. For the variance or spread of the gaussian kernel, namely σ^2, read it from the command line. Try different values and submit your plot for the value that makes most sense to you (observe the projected plots for various spread values and then decide).

Part II: Linear Regression via QR Factorization (50 points)

You will implement the linear regression algorithm via QR factorization, namely Algorithm 23.1 on page 602 in Chapter 23.

CSCI4390: You can use the **numpy.linalg.qr** function to compute the QR factorization.

CSCI6390: You must implement QR factorization on your own, as described in Section 23.3.1 (you cannot use numpy.linalg.qr or similar function).

Next, for both sections, using the $\mathbf{Q}$ and the $\mathbf{R}$ matrices, you must solve for the augmented weight vector $\mathbf{w}$ using backsubstitution. See Example 23.4 on how backsolve works. You cannot use **numpy.linalg.inv** in your solution (but it can be used to verify your answer).

After you have computed the weight vector $\mathbf{w}$, you should compute the SSE value for the predictions, and also the R^2 statistic, defined as:

$$R^2 = \frac{TSS - SSE}{TSS}$$

where TSS is the total scatter of the response variable $TSS = \sum_{i=1}^{n}(y_i - \mu_Y)^2$

What to submit

- Write two python scripts named as **Assign3-kpca.py** and **Assign3-lr.py**, for each of the parts, respectively.
- For part1, read the filename from the command line, assume it is in the local directory. Your script will be run as **Assign3-kpca.py FILENAME ALPHA SPREAD**. FILENAME is the datafile name, ALPHA is the approximation threshold α and SPREAD is the σ^2 parameter for the Gaussian kernel. In other words, your script must compute and return the correct number of components to capture α fraction of total variance. You should also plot the projected points on the first two Kernel PCs.

Note that KPCA on the full dataset with about 20K points will be expensive and it will take around 12GB of memory if you use np.float32 for all data. If you cannot run on the full dataset, run it on the first 10K points. If that is also too much, then you should you use at least 5K points.

- For part2, the script will be run as **Assign3-lr.py FILENAME**, where FILENAME is the input data file. Use 70% of the points for the training dataset, and 30% of the points for the testing dataset. Your script MUST print the following (where MSE is mean squared error):
    1. the weight vector $\mathbf{w}$
    2. the L_2 norm of the weight vector
    3. the SSE, MSE and R^2 values on the training data
    4. the SSE, MSE and R^2 values on the testing data
- Submit a PDF file named Assign3.pdf that should include your answers to each of the questions (just cut and paste the output from python). The figures should also be part of this file. **Failure the submit the PDF will result in lost points.**
- Submit the scripts and pdf file via submitty

Policy on Academic Honesty

You are free to discuss how to tackle the assignment, but all coding must be your own. Please do not copy or modify code from anyone else, including code on the web. Any students caught violating the academic honesty principle will get an automatic F grade on the course and will be referred to the dean of students for disciplinary action.
