SVMs and PCA

For this mini-project (both tasks), you are asked to use Scikit-learn and you are encouraged to use the CoLab cloud service provided by Google for the development and training of ML algorithms. CoLab allows you to run your code using CPU, GPU or TPU runtimes. One exercise in task 1 requires using CoLab.

As CoLab runs your Jupyter Notebook code on cloud infrastructure, your data files need to be accessible to the code. One option is to store the data files on your Google Drive. Follow these simple steps to enable CoLab to access your Google Drive.

Task 1: EigenFaces

"Eigenfaces" is the general name given to a dataset of face images after applying PCA to the images. Eigenfaces are the Principal Components for the dataset visualized as images. Remember that principal components these are the new axes of the transformed dataset in the directions of maximum variance.

Eigenfaces were introduced in a seminal paper by Turk and Pentland in 1991 and have since been used widely as input for classifiers and face recognition systems. The advantage of eigenfaces is lowering the dimensionality of images, saving computational power and **achieving better classification performance.** The combination of SVMs and PCA can be superior to neural networks when the number of images available is not very large.

You may consult this tutorial for an example of classifying face images using PCA and SVMs. The dataset used in this tutorial is the popular **"Labeled faces in the Wild (LFW)"** dataset, still actively in use today. The Vision Research Group at University of Massachusetts keeps a leadboard for best performing recognition algorithms on the LFW dataset.

One of the challenges of face recognition is that faces appear in images in different conditions, such as translation, rotation and luminance, among others. A recognition technique is robust if it can recognize a face despite such conditions.

Please **<u>read</u>** pages 2 and 4 of <u>this paper</u> which discusses the impact of such conditions on the produced PCA and the recognition algorithm.

In the sklearn tutorial provided above:

1. Change the parameter grid for 'C' to include the values (1, 10, 100, 1e3). Lower values of 'C' allows the classifier to handle more difficult datasets, as it can better tolerate misclassified points

2. Now, disable PCA. That is, feed the images from lfw_people directly to the SVM, without applying PCA. Note that you now need to standardize the data (zero mean and unit variance). <u>It's a good idea to use TPU runtime here</u>. Report the best classifier found by the grid search and its performance. <u>Comment on the value of C compared to the best value provided in the tutorial. Justify the difference.</u>

3. Now, go back to using PCA. Vary the number of principal components you use as an input to the classifier [50, 150, 190, 300, 1000]. Plot the number of principal components on the x-axis, and the performance and the corresponding C values found by grid search on the Y-axis.

4. <u>Report on the change in performance you observe.</u> What causes the performance to change with respect to the number of principal components in this particular manner? Recall that omitting low significant principal components causes some loss in the data input to the classifier. Recall also that SVM uses a quadratic solver that is best suited to handling **smaller** datasets.

5. Use the best classifier found by the gridsearch above. Save the eigenface plots produced by the last part in the tutorial using the best configurations (best number of principal components and best classifier parameters).

6. Now, change the dataset to the 'unfunneled' version. Report the change in performance. Here is an extract from LFW about funneling, while the full paper is here.

> There are now four different sets of LFW images including the original and three different types of "aligned" images. The aligned images include "**funneled images**" (ICCV 2007), LFW-a, which uses an unpublished method of alignment, and "deep funneled" images (NIPS 2012). Among these, LFW-a and the deep funneled images produce superior results for most face verification algorithms over the original images and over the funneled images (ICCV 2007).

7. Compare the Eigenfaces plotted in the last part of the tutorial for both the 'funneled' and 'unfunneled' datasets. You will need to observe the corresponding plots closely. It's normal that the luminance differs between the two sets of principal components. What is the other side effect of using unfunneled images on the eigenfaces?

8. Go back to using the best performing classifier and the funneled dataset. Run this classifier using the CPU runtime, the GPU runtime and the TPU runtime. Report the observed speedup if any.

Task 2: Classification of cancer gene expressions

In this task, you are asked to develop an SVM classifier for 14 cancer types according to their gene expressions in tissue samples. This dataset was collected and analyzed using several techniques including SVMs in this paper http://www.pnas.org/content/98/26/15149.full. Please read the Abstract of the paper and the section on SVMs. The results section is an optional reading.

For this task, please use PCA instead of the recursive feature elimination technique used in the paper. The dataset is available for download from: https://web.stanford.edu/~hastie/ElemStatLearn/datasets/14cancer.info.

1- Develop an SVM classifier for the dataset.

    a. Use the train and test datasets provided to calculate training and test scores. This dataset is challenging to solve using general purpose techniques similar to the ones we are using. Expect a test-score of ~70%. 80% test score can be achieved via specialized preprocessing techniques.

    b. The SVM performance on this dataset is particularly sensitive to the number of principal components used. Automate your code to attempt principal component numbers in the range of 90 to125 with step size 5. Plot the training and test scores for the best classifier vs. the number of principal components. That is for each principal component number (For example 95) your code will perform a grid search and calculate both training and test scores. Note that you can use smaller step size around promising points (showing higher performance).

    c. Comment on the generalization or overfitting of the classifier with respect to the number of principal components in the graph above.

2- Report whether you used a one-vs-one or one-versus-rest SVM. Check the SKlearn documentation of SVC.

Bonus

1. Apply hierarchical clustering on the cancer gene expressions dataset used in task 2. Use the homogeneity metric to comment on the quality of clusters produced.

Submission

Your submission should consist of a well-formatted Jupyter notebook that provides Interpretation of each result or visualization.

The notebook should be clear and self-sufficient as it will replace a demo. Maintain the data files with the provided filenames to facilitate testing.

Please submit your Jupyter notebook as an attachment via email to guc.ml18@gmail.com with the subject line (Project 2- team number). The deadline is 11:59PM Saturday 1st December 2018.