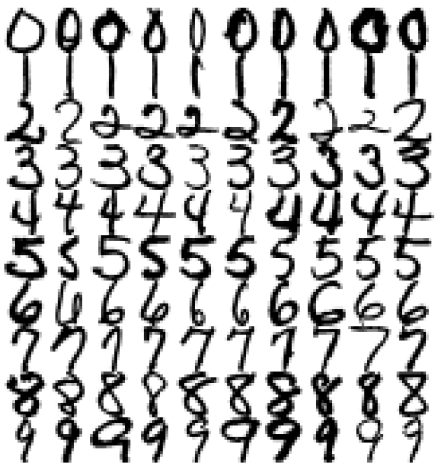
Homework #5

Homework is **due at the start of class** on **Monday, April 29, 2019**, without exceptions unless permission was obtained from the instructor **in advance**.
You may collaborate with other students, discuss the problems and work through solutions together. However, you **must write up your solutions on your own**, without copying another student's work or letting another student copy your work. In your solution for each problem, you must write down the names of any person with whom you discussed it.
This will **not** affect your grade.

1. (**Principal Component Analysis**, 50 points) We explore how to use PCA to generate "good" features for Handwritten Digit Recognition using the USPS data set. A reduced and pre-processed version of this data set is available on eLearning with files `usps.train`, `usps.valid` and `usps.test`. The files are in CSV format, with the first column being the digit label and the next 256 columns representing gray scale intensities of the 16×16 digit. The features are all in the range $[0, 1]$. Perform the following experiment and write a brief report with visualizations, plots and answers. Denote the original data set X_{100}, which corresponds to using all the features to capture 100% of the variance, and let $k_{100} = 256$ denote the total dimension of the original data set.



a. Perform PCA on the training data and **visualize** the top 16 *eigendigits*.

b. **Plot** the cumulative explained variance ratio vs. number of components. What dimensionality does it take to achieve 70%, 80% and 90% of the total variance? Denote these lower dimensionalities k_{70}, k_{80} and k_{90}.

c. Use `sklearn.linear_model.SGDClassifier` with settings `loss='hinge'` and `penalty='l2'` to realize a linear support vector machine classifier optimized via stochastic gradient descent. This is a 10-class classification problem, and `SGDClassifier` supports multi-class classification by combining binary classifiers in a *one-vs-all* (OVA) scheme[1]. For each of the three projections and the original data set (i.e., $k = k_{70}$, k_{80}, k_{90}, k_{100}), perform the following steps:

   (a) Compute the projection X_f, by projecting the data on to the top k_f eigenvectors, where $f = 70$, 80, 90. For $f = 100$, simply use the original training set.

   (b) Learn different multi-class SVM classifiers for each $\alpha \in \{0.0001, 0.001, 0.01, 0.1\}$. α corresponds to the weight on the regularization term and can be passed to `SGDClassifier` via `alpha=0.001`.

   (c) Evaluate the learned SVM model on the **validation set**.

   Report the validation error of each (k_f, α) as a table. Note that k_{100} corresponds to the original data set.

d. **Report** the error of the best (k, α) pair on the **test data**? **Explain** how it compares to the performance of the SVM without feature selection?

[1] http://scikit-learn.org/stable/modules/sgd.html

2. (**Spectral Clustering**, 50 points) In this problem, we will take a look at a clustering algorithm based on the eigenvalues of a matrix, referred to as **spectral clustering**. Given data points $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$, we construct a matrix $A \in \mathbb{R}^{n \times n}$ of similarities and analyze its eigenvalues/vectors. Perform the following experiment and write a brief report with visualizations, plots and answers.

a. Generate a noisy, synthetic data set:

```
from sklearn import datasets
x = datasets.make_circles(n_samples=1500, factor=.5, noise=.05)
```

b. Write a function `spectral_clustering(x, k)` that implements the following steps:

- Construct a **symmetric** $n \times n$ matrix of pairwise similarities between all data points such that $A_{ij} = A_{ji} = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$ for $i, j = 1, \ldots, n$ and $\gamma > 0$. This step can be very slow for even moderate data sets. Consider using `scipy.spatial.distance.pdist`[2].
- Compute the **Laplacian matrix**, $L = D - A$, where D is a diagonal matrix with $D_{ii} = \sum_{j=1}^n A_{ij}$ (the row sum of A) for all $i = 1, \ldots, n$; that is, D is a diagonal matrix of the row sums of A.
- Compute the eigenvalues and eigenvectors of L, and select eigenvectors V_k corresponding to the k **smallest** eigenvalues of L.
- Let the rows of $V_k \in \mathbb{R}^{n \times k}$ be $\mathbf{v}_i \in \mathbb{R}^k$. These rows are a **lower-dimensional representation** of the training examples $\mathbf{x}_i \in \mathbb{R}^n$. Use `sklearn.cluster.KMeans`[3] to cluster the rows $\mathbf{v}_i$ into clusters $C_1, \ldots, C_k$.
- Generate the clustering output such that $\mathbf{x}_i \in C_j$ if $\mathbf{v}_i \in C_j$, for $j = 1, \ldots, k$.

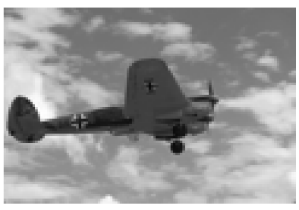
c. Use `sklearn.cluster.KMeans` directly to compute an alternative clustering for the two-dimensional circles data set generated above. **Generate** a scatter plot of the clusters, where the points are colored by the cluster they belong to.

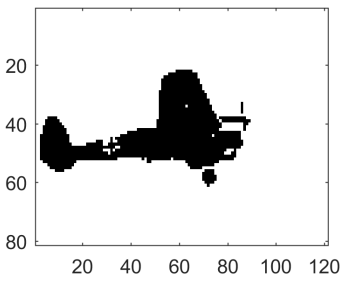
d. Use your function `spectral_clustering(x, k)` to compute the clustering for the two-dimensional circles data set generated above with $k = 2$ and different values of γ. Find a value of γ such that spectral clustering outperforms k-means clustering. **Generate** a scatter plot of the clusters, where the points are colored by the cluster they belong to.

e. We can use spectral clustering to **segment images**. Load the 81×121 image `seg.jpg`.

```
import cv2
img = cv2.imread('seg.jpg', 0)
cv2.imshow('image', img)
```

- We consider each pixel as a single data point and construct a similarity matrix for pairs of pixels, resulting in an adjacency matrix of size 9801×9801. Care should be taken while setting γ.
- Perform the same comparison between k-means and spectral clustering as before with $k = 2$. **Visualize** the segmented images produced by both approaches. **Do not perform a full eigen-decomposition of the Laplacian matrix as it will be very time-consuming.**





[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html
[3] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html