MAT 167: Homework Assignment #8 (due Wednesday, June 6)

First of all, do the following:

- Read Chapters 9, 10, & 11.

Problem 1. Using MATLAB, do the following handwritten digit recognition experiments.

(a) Download the handwritten digit database from the following link:
http://www.math.ucdavis.edu/~saito/courses/167.s12/usps.mat . Then, load this file into your MATLAB session. This file contains 4 arrays: `train_patterns`, `test_patterns` of size 256×4649, and `train_labels`, `test_labels` of size 10×4649. The `train_patterns` and `test_patterns` contain a raster scan of the 16×16 gray level pixel intensities, which have been normalized to range within $[-1, 1]$. The `train_labels` and `test_labels` variables contain the ground truth information of the digit images. That is, if the jth handwritten digit image in `train_patterns` truly represents digit i, then the $(i+1, j)$th entry of `train_labels` is +1, and all the other entries of the jth column of `train_labels` are -1.

Now, display the first 16 images in `train_patterns` using `subplot(4,4,k)` and `imagesc` functions in MATLAB. Print out the figure and attach it to your hw8 submission.

[Hint: You need to `reshape` each column into a matrix of size 16×16 followed by transposing it in order to display it correctly.]

(b) Now, compute the mean digits in the `train_patterns`, put them in a matrix called `train_aves` of size 256×10, and display these 10 mean digit images using `subplot(2,5,k)` and `imagesc`. Print out the figure and attach it to your hw8 submission.

[Hint: You can gather (or pool) all the images in `train_patterns` corresponding to digit $k-1$ ($1 \leq k \leq 10$) by the following:

```
>> train_patterns(:, train_labels(k,:)==1);
```

]

(c) Let's conduct the simplest classification experiments as follows:

(c.1) First, prepare a matrix called `test_classif` of size 10×4649 and fill this matrix by computing the Euclidean distance (or its square) between each image in the `test_patterns` and each mean digit image in `train_patterns`.

[Hint: the following line computes the squared Euclidean distances between all the test digit images and the kth mean digit of the training dataset by one line:

```
>> sum((test_patterns-repmat(train_aves(:,k),[1 4649])).^2);
```

]

(c.2) Then, compute the classification results by finding the position index of the minimum of each column of `test_classif`. Put the results in a vector `test_classif_res` of size 1×4649.

[Hint: You can find the position index giving the minimum of the jth column of `test_classif` by

```
>> [tmp, ind] = min(test_classif(:,j));
```

Then, the variable `ind` contains the position index (between 1 and 10) of the smallest entry of `test_classif(:,j)`.
]

(c.3) Finally, compute the confusion matrix `test_confusion` of size 10×10, print out this matrix, and submit your results.

[Hint: First gather the classification results corresponding to the kth digit by

```
>> tmp=test_classif_res(test_labels(k,:)==1);
```

This `tmp` array contains the results of your classification of the test digits whose true digit is $k-1$ ($1 \leq k \leq 10$). In other words, if your classification results were perfect, all the entries of `tmp` would be k. But in reality, this simplest classification algorithm makes mistakes, so `tmp` contains values other than k. You need to count how many entries have the value j in `tmp`, $j = 1 : 10$. That would give you the kth row of the `test_confusion` matrix.]

(d) Finally, let's conduct the SVD-based classification experiments.

(d.1) Pool all the images corresponding to the kth digit `train_patterns`, compute the rank 17 SVD of that set of images (i.e., the first 17 singular values and vectors), and put the left singular vectors (or the matrix U) of kth digit into the array `train_u` of size $256 \times 17 \times 10$. For $k = 1 : 10$, you can do the following:

```
>> [train_u(:,:,k),tmp,tmp2] = svds(train_patterns(:,train_labels(k,:)==1),17);
```

We do not need the singular values and right singular vectors in this experiment.

(d.2) Now, compute the expansion coefficients of each test digit image with respect to the 17 singular vectors of each train digit image set. In other words, you need to compute 17×10 numbers for each test digit image. Put the results in the 3D array `test_svd17` of size $17 \times 4649 \times 10$. This can be done by

```
>> for k=1:10
     test_svd17(:,:,k) = train_u(:,:,k)' * test_patterns;
   end
```

(d.3) Next, compute the error between each original test digit image and its rank 17 approximation using the kth digit images in the training dataset. The idea of this classification is that if a test digit image should belong to class of k^*th digit if the corresponding rank 17 approximation is the best approximation (i.e., the smallest error) among 10 such approximations. (See my Lecture 21 for the details). Prepare a matrix `test_svd17res` of size 10×4649, and put those approximation errors into this matrix.

[Hint: The rank 17 approximation of test digits using the 17 left singular vectors of the kth digit training images can be computed by `train_u(:,:,k)*test_svd17(:,:,k);`]

(d.4) Finally, compute the confusion matrix using this SVD-based classification method by following the same strategy as Parts c.2 and c.3 above. Let's name this confusion matrix `test_svd17_confusion`. Print out this matrix, and submit your results.

Problem 2: Let $A \in \mathbb{R}^{m \times n}$, $W \in \mathbb{R}^{m \times k}$, and $H \in \mathbb{R}^{k \times n}$. Suppose we know the values of entries of A and H, and want to determine the values of entries of W by the least squares, i.e., find H that minimizes $\|A - WH\|_F^2$. Then, show that this minimization leads to the following version of the normal equation:

$$HH^\mathsf{T} W^\mathsf{T} = HA^\mathsf{T}.$$

Problem 3: Using MATLAB, do the following text mining experiments.

(a) Download the NIPS dataset from the following link:
http://www.math.ucdavis.edu/~saito/courses/167.s12/nips.mat . Then, load this file into your MATLAB session. This file contains a term-document matrix A of size 12419×1500 as discussed in Lecture 22. Actual 12,419 terms are included in an array `terms` in that file. Suppose you want to retrieve the documents containing the following three terms: 'principal', 'component', 'analysis'. Construct the query vector $\boldsymbol{q}$ in MATLAB.

[Hint: Use `strcmp` function of MATLAB to get the position of each term you want to use in the query in the array `terms`.]

(b) Now, compute the cosine similarities between this query vector $\boldsymbol{q}$ and each document (i.e., each column vector) $\boldsymbol{a}_j$, $j = 1 : 1500$. Then, plot this cosine similarities and submit your plot. Also, compute the number of retrieved documents by varying the tolerance $\mathrm{tol} = 0.05, 0.15, 0.25, 0.35$. Report these four numbers retrieved.

(c) Compute the first 100 terms of SVD of A using MATLAB's `svds` function by:

```
>> [U100, S100, V100] = svds(A, 100);
```

Then, compute the relative Frobenius error between A_{100} and A, and report the results.

(d) Instead of A, let's use the rank 100 approximation of A. Without forming A_{100} explicitly, repeat Part (b) using the cosine similarity formula discussed in the class, i.e.,

$$\cos\theta_j := \frac{\boldsymbol{q}_k^{\mathsf{T}}{}_j}{\|\boldsymbol{q}\|_2 \|{}_j\|_2}, \quad \boldsymbol{q}_k := U_k^{\mathsf{T}} \boldsymbol{q}.$$

(e) Instead of $k = 100$ in Parts (c) and (d), what happens if you use $k = 50$? Repeat these parts using the rank 50 approximation, and discuss the difference between $k = 50$ and $k = 100$.