Large-Scale Machine Learning EECS 6898, Homework 1

October 7, 2010

Notes:
The due date is 23:59, Oct. 19, 2010.
20 points in total. 2 points would be dropped for each late day.
Feel free to discuss with anyone. But you should answer the questions, complete the programming implementations, and write the report totally on your own.

1 Review Exercises (4 points)

1.1 Johnson-Lindenstrauss(JL) lemma (2 points)

In the lecture slides, one side of Johnson-Lindenstrauss(JL) lemma, i.e., $||f(u)-f(v)||_2^2 \leq (1+\varepsilon)||u-v||_2^2$, was proved. Prove the other side: $||f(u)-f(v)||_2^2 \geq (1-\varepsilon)||u-v||_2^2$

1.2 Spectral clustering (1 point)

W is a symmetric matrix, and y is a vector, prove that $\frac{1}{2}\sum\limits_{i,j} W_{ij}||y_i - y_j||_2^2 = y^T(D-W)y$, where D is a diagonal matrix, $D_{i,i} = \sum\limits_j W_{i,j}$.

1.3 Low rank approximation (1 point)

Suppose A_k is the best k-rank approximation of matrix A, i.e., $A_k = \min\limits_D ||A - D||_F, s.t., rank(D) = k$. Prove that $||A-A_k||_F^2 = \sum\limits_{i=k+1}^{n} \sigma_i^2$ and $||A-A_k||_2 = \sigma_{k+1}$, where σ_i is the i-th largest singular value of A.

2 Programming assignments (16 points)

Please write an experimental report with your results, observations, etc., together with a brief (a couple sentences) description of each module in your code.
Your submitted file should be named as "hw1_uni_name.zip", e.g., "hw1_jh2700_Junfeng.zip",

which contains all your code and your report. You can use any programming language for the experiments, but Matlab (especially 64-bit Matlab) is highly recommended, which would really make your life much easier.

You are given 2 datasets, each of which contains 7000 points in Matlab format. For every step in the following tasks, you are supposed to use both datasets.

Notes: Each point in the first dataset is a 512 dimension gist feature extracted from a web image. For the intuitive understanding, the corresponding images are also provided. Each point in the second data set is a 784 dimension vector, representing the $28 * 28 = 784$ pixels for a handwritten digit image. You can use matlab commands like
im=reshape(x, 28,28);imshow(im);
to see the handwritten digit image, where x is one 784 dimension vector.

2.1 Sparse Matrix (4.5 points)

2.1.1 (0.5 point)

Create a dense graph $W_{ij} = \exp(-||x_i - x_j||^2/\sigma^2)$. Here, the parameter σ is used as the mean of pairwise distance, i.e., $\sigma^2 = \frac{1}{N^2} \sum_{i,j=1}^{N} ||x_i - x_j||^2$.
Create a sparse graph: $W_{ij} = \exp(-||x_i - x_j||^2/\sigma^2)$, if point j is point i's top 10 nearest neighbors or point i is point j's top 10 nearest neighbors; $W_{ij} = 0$, otherwise.

2.1.2 (1 point)

Apply spectral clustering on dense and sparse W, get the reduced 10 dimensions (hint: ignore the eigen vector(s) with 0 eigen value).

2.1.3 (1 point)

Apply K-means clustering to get 10 clusters based on the reduced 10 dimensions. Show sample images from each of the 10 clusters for both dense and sparse cases.

2.1.4 (1 point)

Apply regular SVD and Sparse SVD (via Lanczos method) on the sparse W, and compare the accuracy of top 10 eigenvalues, eigenvectors and time taken.

2.1.5 (1 point)

Normalize the sparse W such that it becomes a transition matrix, i.e., sum of each row = 1. Apply page rank with power method on the normalized W to find the stationary probability distribution for each point. Try different α , for example, $\alpha = 0, 0.001, 0.1$ in your experiments. Show the top 20 images with the highest stationary probability distribution.

2.2 Randomized SVD (1.5 points)

2.2.1 (0.5 point)

On dense W, apply exact SVD.

2.2.2 (1 point)

On dense W, apply randomized SVD with l columns samples. $l = 100, 120, 150, 200$. Compare the eigenvalues, eigenvectors and speed with exact SVD for $k = 100$.

2.3 Approximate matrix multiplication (2 points)

With the dense W, apply the sampling based approximation of matrix multiplication for WW^T. Report the errors compared to exact multiplication, i.e., $||AB - CF||_F$, where $A = W, B = W^T$, with l columns sampled. Try $l = 100, 200, 400, 800$. Also compare the corresponding speed in comparison with the exact method.

2.3.1 (1 point)

Use uniform sampling.

2.3.2 (1 point)

Use Sampling based on the column norm.

2.4 Low rank approximation with column sampling and Nystrom method (5 points)

With the dense W, apply column sampling and Nystrom method for spectral reconstruction and matrix projection, with l columns sampled. Use uniform sampling. Try $l = 100, 200, 400, 800$. (Hint: You are supposed to show figures as in slides 31-33 of "Matrix Approximatins II"). Also compare the speeds for different methods.

2.4.1 (2 points)

Compare the top 20 eigen values and eigen vectors obtained by these methods with the ones obtained via exact SVD.

2.4.2 (2 points)

For k=100, compute the relative accuracy for k-rank approximation matrix $\widetilde{G}_k$: $\frac{||G-\widetilde{G}_k||_F}{||G-\widehat{G}_k||_F}$. Note that for spectral reconstruction, $\widetilde{G}_k = \widetilde{U}_k\widetilde{\Sigma}_k\widetilde{U}_k^T$, for matrix projection $\widetilde{G}_k = \widetilde{U}_k\widetilde{U}_k^T G$. $\widehat{G}_k$ is obtained based on exact SVD, i.e., $\widehat{G}_k =$

$\widehat{U}_k \widehat{\Sigma}_k \widehat{U}_k^T$, and $\widehat{G}_k = \widehat{U}_k \widehat{U}_k^T G$ for spectral reconstruction and matrix projection respectively.

2.4.3 (1 point)

Generate a random dataset with 512 dims and 7000 pts (each entry sampled from U[0, 1]). Compare the above quantities for that matrix.

2.5 Sampling (3 points)

Redo the problem 2.4 with adaptive sampling with batch size of 10 columns per iteration. Compare the performance and speed with uniform sampling.