The University of Texas at Austin
Department of Electrical and Computer Engineering

EE381V: Large Scale Learning — Spring 2013

Assignment 1

Caramanis/Sanghavi   Due: Thursday, Feb. 7, 2013.

(Problems 1 and 2 have been adapted from Jure Leskovec's course on 'Mining Massive Datasets')

1. **Locality Sensitive Hashing**

   The first week of class discussed locality sensitive hashing. For some standard similarity functions, like the Jaccard similarity, we showed that there corresponds a locality sensitive hashing scheme. As it turns out, not all similarity functions have a locality sensitive hashing scheme. This problem explores this issue.

   Recall that a locality sensitive hashing scheme is a set F of hash functions that operate on a set S of objects, such that for two objects $x, y \in S$,

$$\Pr_{h \in \mathcal{F}} [h(x) = h(y)] = \mathrm{sim}(x, y)$$

   where $\mathrm{sim}(\cdot) : S \times S \to [0, 1]$ is a pairwise function (the similarity function).

   - (5 points) Let $d(x, y) = 1 - sim(x, y)$. Prove that for $\mathrm{sim}(\cdot)$ to have a locality sensitive hashing scheme, $d(x, y)$ should satisfy the triangle inequality.

$$d(x, y) + d(y, z) \geq d(x, z)$$

     for all $x, y, z \in S$.

   - (5 points) Consider the following two similarity functions: The so-called Overlap similarity function,

$$\mathrm{sim}_{Over}(A, B) = \frac{|A \bigcap B|}{\min(|A|, |B|)}$$

     and the Dice similarity function,

$$\mathrm{sim}_{Dice}(A, B) = \frac{2|A \bigcap B|}{(|A| + |B|)},$$

     where A and B are two sets.

     Is there a locality sensitive hashing scheme for either? Prove, or disprove, giving a counterexample.

     Prove or disprove (give counterexamples). Let A, B be any two sets.

Solution: Let $x, y, z \in S$. Then,

$$\begin{aligned}
P(h(x) \neq h(y)) &= P(h(x) \neq h(y), h(x) = h(z)) + P(h(x) \neq h(y), h(x) \neq h(z)) \\
&\leq P(h(y) \neq h(z)) + P(h(x) \neq h(z)) \\
1 - \text{sim}(x, y) &\leq 1 - \text{sim}(y, z) + 1 - \text{sim}(x, z) \\
d(x, y) &\leq d(y, z) + d(z, x)
\end{aligned}$$

Now let $A, B, C \subseteq \{1, 2, \ldots, 10\}$. $A = \{1, 2, 3, 4, 5\}$, $B = \{1, 2, 6, 7, 8\}$, $C = \{1, 2, 3, 4, 6, 8\}$. Then, $\text{sim}_{Over}(A, B) = \frac{2}{5}$, $\text{sim}_{Over}(B, C) = \frac{4}{5}$, $\text{sim}_{Over}(A, C) = \frac{4}{5}$.

Then,

$$d(A, B) = 1 - \frac{2}{5} = \frac{3}{5} > \frac{2}{5} = (1 - \frac{4}{5}) + (1 - \frac{4}{5}) = d(A, C) + d(B, C)$$

Hence $\text{sim}_{Over}()$ cannot have a LSH scheme. The same example can be used to show that $\text{sim}_{Dice}()$ function also cannot have a LSH scheme.

2. **Approximate Near Neighbor Search using LSH**

LSH has been used for nearest neighbor search, in numerous applications. This problems explores this.

Given a data set $\mathcal{A}$, along with a distance function $d(\cdot)$, the Nearest Neighbor problem says the following: given a query point, z, return its nearest neighbors, w.r.t. $d(\cdot)$. The approximate nearest neighbor problem is an approximation in two respects: it only requires us to know the immediate neighborhood of any given point, and also, we need only return approximate nearest neighbors, up to a dilation factor λ. More precisely, the (c, λ)-Approximate Near Neighbor (ANN) problem is as follows: Given a query point, z, for which (we assume) there exists a point $x \in \mathcal{A}$ with $d(x, z) \leq \lambda$, return a point x' from the dataset with $d(x', z) \leq c\lambda$.

We outline an approximate nearest neighbor algorithm, and then, through the parts of this problem, show that with large probability, it outputs a c-approximate nearest neighbor, as explained above.

Let $\mathcal{A}$ be a dataset with n points from a metric space with distance measure $d()$. Let $\mathcal{H}$ be a $(\lambda, c\lambda, p_1, p_2)$ locally sensitive family of hash functions for the distance measure $d()$. Let $\mathcal{G} = \mathcal{H}^k = \{g = (h_1, \ldots, h_k) | h_i \in \mathcal{H}\}$, where $k = \log_{1/p_2}(n)$ be an amplified family. Choose $L = n^\rho$ random members $g_1, \ldots, g_L \in \mathcal{G}$, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$. We then do the following: (a) hash all the data points and the query point z using all g_i's ($1 \leq i \leq L$); (b) retrieve at most $3L$ data points from the buckets $g_j(z)$ ($1 \leq j \leq L$), and (c) report the closest one as a (c, λ)-ANN.

- (5 points) Define $W_j = \{x \in \mathcal{A} | g_j(x) = g_j(z)\}$ ($1 \leq j \leq L$) as the random set of data points x hashed to the same bucket as the query point z by the hash function g_j. Let $T = \{x \in \mathcal{A} | d(x, z) > c\lambda\}$. Prove:

$$Pr\left[\sum_{j=1}^{L} |T \bigcap W_j| > 3L\right] < \frac{1}{3}.$$

- (5 points) Let $x^* \in \mathcal{A}$ be a data point such that $d(x^*, z) \leq \lambda$. Prove:

$$Pr\left[g_j(x^*) \neq g_j(z), \forall 1 \leq j \leq L\right] < \frac{1}{e}.$$

- (5 points) Find a bound on δ, the probability that the reported point is an actual (c, λ)-ANN.

Solution: Let $|T| = m \leq n$. Suppose $x \in T$. Then,

$$P(g(x) = g(z)) = P(h(x) = h(z))^k \leq p_2^k = p_2^{\log_{1/p_2} n} = \frac{1}{n}$$

Hence $E(|T \bigcap W_j|) = mP(g(x) = g(z)) = \frac{m}{n} < 1$. Now using Markov's inequality

$$P\left[\sum_{j=1}^{L} |T \bigcap W_j| > 3L\right] \leq \frac{E\left[\sum_{j=1}^{L} |T \bigcap W_j| > 3L\right]}{3L} = \frac{E(|T \bigcap W_j|)L}{3L} < \frac{1}{3}$$

Now,

$$\begin{aligned}
P\left[g_j(x^*) \neq g_j(z), \forall 1 \leq j \leq L\right] &= \left(P(g_1(x^*) \neq g_1(z))\right)^L \\
&= \left(1 - P(g_1(x^*) = g_1(z))\right)^L \\
&\leq (1 - p_1^k)^L \\
&= \left(1 - p_1^{\log_{1/p_2} n}\right)^L \\
&= (1 - n^{-\rho})^L \\
&\leq e^{-\frac{L}{n^\rho}} = e^{-1} \quad (\text{using } 1 - x \leq e^{-x})
\end{aligned}$$

Let $\mathcal{E}$ be the error event. Suppose $x^* \in A$, $d(x^*, z) < \lambda$. Now error may occur if, (a) there are more than $3L$ points in the L buckets $g_j(z)$ and all $3L$ sampled points are from set T (call this event E_1), (b) the point x^* is not hashed in any of the L buckets $g_j(z)$ (call this event E_2). Note that error cannot happen if the event $(E_1 \bigcup E_2)^c$ occurs. The probability of error can be bounded as follows,

$$\begin{aligned}
P(\mathcal{E}) &= P(\mathcal{E}, E_1 \bigcup E_2) \leq P(\mathcal{E}, E_1) + P(\mathcal{E}, E_2) \leq P(E_1) + P(E_2) \\
&= P\left[\sum_{j=1}^{L} |T \bigcap W_j| > 3L\right] + P\left[g_j(x^*) \neq g_j(z), \forall 1 \leq j \leq L\right] \\
&\leq \frac{1}{3} + \frac{1}{e} = \epsilon \;\; (say)
\end{aligned}$$

Hence we can guarantee that the reported point is an actual (c, λ)-ANN with probability greater than $\delta = 1 - \epsilon$.

3. This problem tests empirically how nearest-neighbor search using LSH compares to linear search. For this, we provide a dataset of images. Each column in the dataset is a vectorized 20×20 image patch. Download the image set and matlab code here: `http://http://users.ece.utexas.edu/~cmcaram/LSL2013/lsh.zip`[1] and see the `ReadMe.txt` file for instruction on using the code, and in particular, the functions `lsh` and `lshlookup`.

   In this problem we use the ℓ_1 distance measure. The LSH function is run with $L = 10, k = 24$, where L is the number of hash tables generated and k is the length/number of bits of the hash key.

   - (10 points) Consider the image patches z_j, of column $100 \times j$, for $j \in \{1, 2, \ldots, 10\}$. Find the top 3 nearest neighbors for these image patches, (excluding the original patch) using both LSH and linear search.

     Compare the average search time for LSH and linear search. (If the bucket contains less than 3 points, rehash till you get enough neighboring points).
   - (10 points) For each z_j $(1 \leq j \leq 10)$ let $\{x_{ij}\}_{i=1}^{3}$ denote the approximate near neighbors of z_j found using LSH, and $\{x_{ij}^{*}\}_{i=1}^{3}$ to be the actual top 3 near neighbors of z_j found using linear search. Compute the following error measure:

     $$error = \frac{1}{10} \sum_{j=1}^{10} \frac{\sum_{i=1}^{3} d(x_{ij}, z_j)}{\sum_{i=1}^{3} d(x_{ij}^{*}, z_j)}$$

     Plot the error value as a function of L (for $L = 10, 12, 14, \ldots, 20$, with $k = 24$). Then plot the error values as a function of k (for $k = 16, 18, 20, 22, 24$ with $L = 10$).
   - (5 points) Plot the top 10 near neighbors found using the two methods (using the default $L = 10, k = 24$) for the image patch in column 100, together with the image patch itself. Use functions *reshape*() and *mat2gray*() to convert the matrices to images, then use the functions *imshow*() and *subplot*() to display the images. Compare them visually.

   Solution: The relevant plots are shown in Figure 1, 2 and 3.

4. **K-Mean vs. EM for Gaussian Mixture Models**

   Consider samples being generated from a Gaussian mixture model with the following pdf

   $$\begin{aligned} p(x|z=k) &= \frac{1}{2\pi|\Sigma_k|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\right], \quad k \in \{1, \ldots, K\} \\ p(x) &= \frac{1}{K} \sum_{k=1}^{K} p(x|z=k) \end{aligned}$$

   where $x, \mu_k \in \mathbb{R}^2$, $z \in \{1, \ldots, K\}$, $K = 3$, $\Sigma_k \in \mathbb{R}^{2 \times 2}$ are the covariance matrices.

   Let $\mu_1 = [5, 5]^T, \mu_2 = [10, 20]^T, \mu_3 = [16, 8]^T$ and

[1] This is the same as that provided from the problem's source, Jure Leskovec's course on Mining Massive Data Sets. The dataset and code are adapted from Brown University's Greg Shakhnarovich

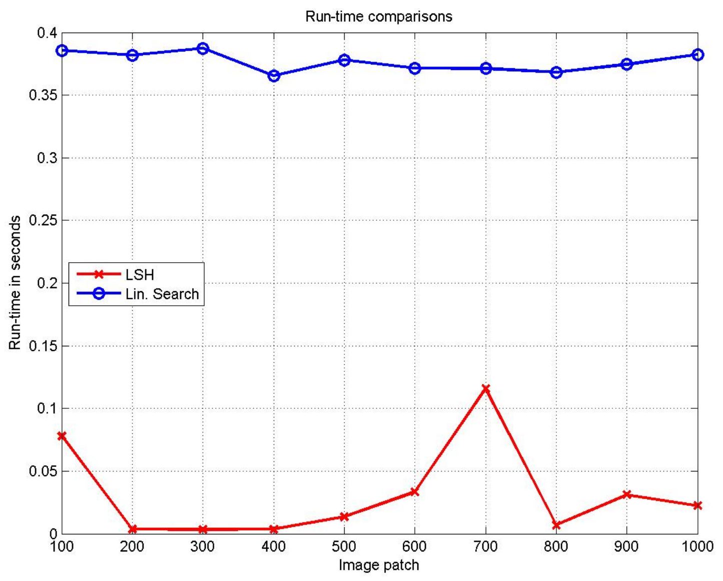
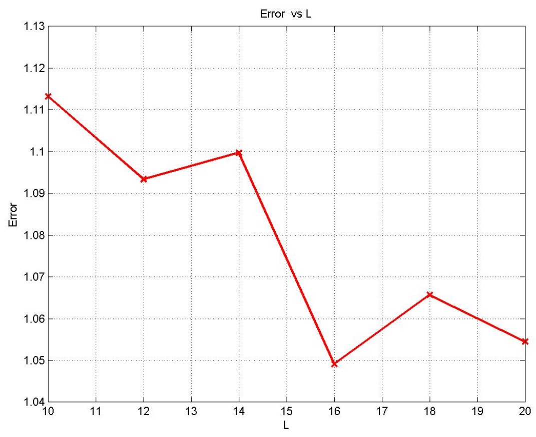
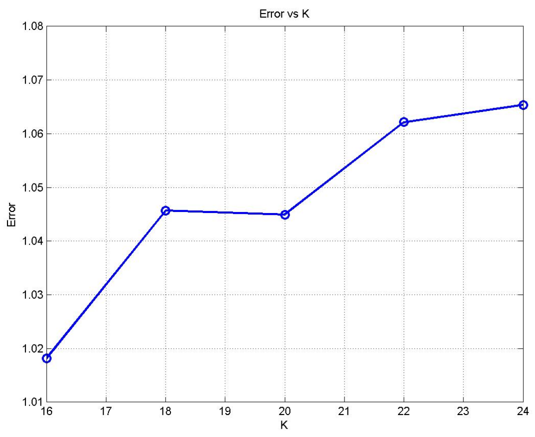


Figure 1: Runtime comparison between linear search and LSH based search of top 3 nearest neighbors. The LSH based search is much faster then the linear search.

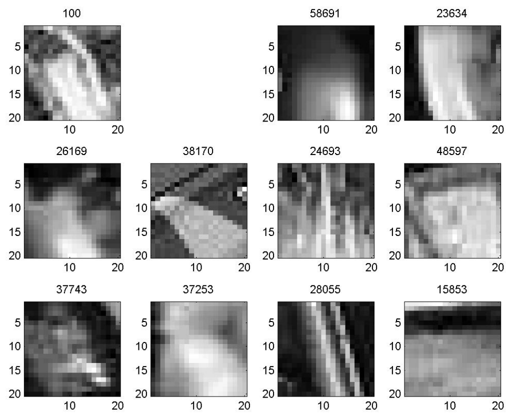


(a) Error vs L



(b) Error vs K

Figure 2: Error variation with L and K. We see that the error approximately reduces with increasing L and increases with increasing K as expected. This is because by increasing the number of hash tables L the chances of the actual nearest neighbors falling in the same bucket as that of the query point increases, hence the error decreases. Where as with increasing key length K the total number of buckets increases, hence the chances of true neighbors falling in the same bucket as the query point decreases, increasing the error.


(a) Actual top 10 nearest neighbors

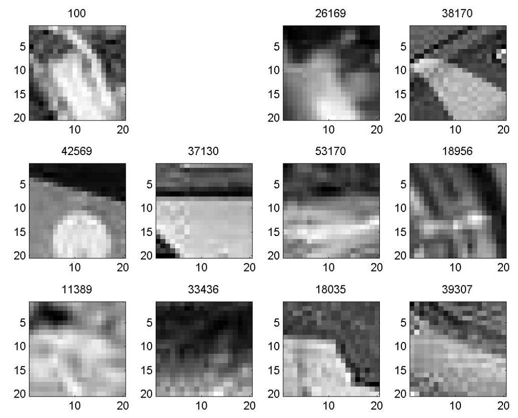

(b) Top 10 ANN returned by LSH based search

Figure 3: Visual comparison between actual top 10 nearest neighbors and those obtained by LSH

$$\Sigma_1 = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \Sigma_2 = \sigma^2 \begin{bmatrix} 1.2 & 0 \\ 0 & 1.2 \end{bmatrix}, \quad \Sigma_3 = \sigma^2 \begin{bmatrix} .28 & .42 \\ .42 & 1.4 \end{bmatrix}$$

- (10 points) We want to cluster the points generated from the distribution $p(x)$ using the EM algorithm. Generate a sample set $S = \{x_i\}_{i=1}^n$ of size $n = 300$ from the distribution $p(x)$, using $\sigma^2 = 2$. Starting with initial estimates of the means as $\widehat{\mu}_1^{(0)} = [0,0]^T, \widehat{\mu}_2^{(0)} = [10,15]^T, \widehat{\mu}_3^{(0)} = [16,0]^T$ and covariance $\widehat{\Sigma}_k^{(0)} = I$, the 2×2 identity matrix, find the final estimates of the means $\widehat{\mu}_k$ and covariances $\widehat{\Sigma}_k$, $k \in \{1,2,3\}$ of the three components of $p(x)$ using the EM algorithm. Now using the MAP estimates $\hat{z}_i = \arg\max_k P(z_i = k|S)$, obtained by the EM algorithm, cluster the sample set S in 3 clusters. Plot the clusters using the matlab function *scatter*() using separate colors for each cluster (or plot each cluster in separate graphs).

- (10 points) Define the set $E_\pi = \{x_i : \pi(\hat{z}_i) \neq z_i\}$, for a particular permutation π of the labels of the clusters. Define error set $E = E_{\pi_0}$, where $\pi_0 = \arg\min_\pi |E_\pi|$. Hence set E contains the sample points that fall in the wrong cluster. The error fraction is given by $e = \frac{|E|}{n}$. Define the probability of error $P_A(E, \sigma^2)$ for a particular clustering algorithm A as the average e over several sample sets S drawn from the same distribution $p(x)$. Now generate several sample sets varying σ^2 between 1 and 30 (also many sample sets for each σ^2) and cluster using both EM and K-mean algorithms starting with same set of initial mean estimates as given in part (a). Plot the probability of error $P_A(E, \sigma^2)$ as a function of σ^2 for both the algorithms. Also plot the average run-time of both the algorithms for different σ^2.

- (5 points) Define the algorithm B as follows. First run the K-mean algorithm to obtain mean estimates $\hat{\mu}'_k$. Then run the EM algorithm using $\hat{\mu}'_k$ as the initial mean estimates. Plot probability of error $P_B(E, \sigma^2)$ vs. σ^2 for algorithm B.

Solution: Let $\hat{z}_{i,k} = P(Z_i = k|x)$ and $p_k = P(Z_i = k)$ be the priors (can be initialized to $1/K$). Then the update equations for EM are as follows.

- *E step:* Calculate the MAP estimates.

$$\hat{z}_{i,k} = \frac{p_k \frac{1}{|\widehat{\Sigma}_k|^{1/2}} \exp\left[-\frac{1}{2}(x_i - \hat{\mu}_k)^T \widehat{\Sigma}_k^{-1}(x_i - \hat{\mu}_k)\right]}{\sum_{j=1}^{K} p_j \frac{1}{|\widehat{\Sigma}_j|^{1/2}} \exp\left[-\frac{1}{2}(x_i - \hat{\mu}_j)^T \widehat{\Sigma}_j^{-1}(x_i - \hat{\mu}_j)\right]}$$

- *M step:* Estimate parameters

$$\begin{aligned} n_k &= \sum_{i=1}^{n} \hat{z}_{i,k} \\ \hat{\mu}_k &= \frac{1}{n_k}\sum_{i=1}^{n} \hat{z}_{i,k} x_i \\ \widehat{\Sigma}_k &= \frac{1}{n_k}\sum_{i=1}^{n} \hat{z}_{i,k}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T \\ p_k &= \frac{n_k}{n} \end{aligned}$$

For $\sigma^2 = 2$ the estimated means and variances were.

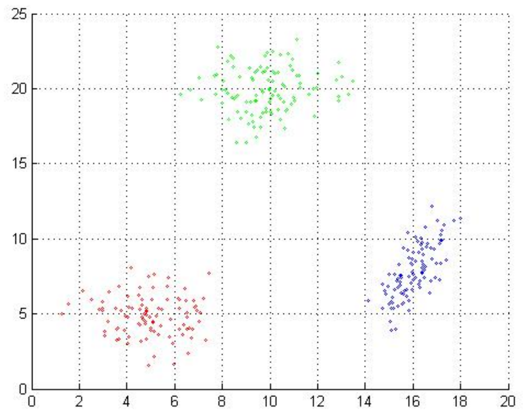
$$\begin{aligned} \hat{\mu}_1 &= [4.9759\ 4.8925] \quad \hat{\mu}_2 = [9.8509\ 19.8908] \quad \hat{\mu}_3 = [16.0278\ 7.8480] \\ \widehat{\Sigma}_1 &= \begin{bmatrix} 1.7987 & -0.0990 \\ -0.0990 & 1.6604 \end{bmatrix} \quad \widehat{\Sigma}_2 = \begin{bmatrix} 2.0729 & 0.1839 \\ 0.1839 & 2.0478 \end{bmatrix} \quad \widehat{\Sigma}_3 = \begin{bmatrix} 0.5635 & 0.9197 \\ 0.9197 & 2.9577 \end{bmatrix} \end{aligned}$$

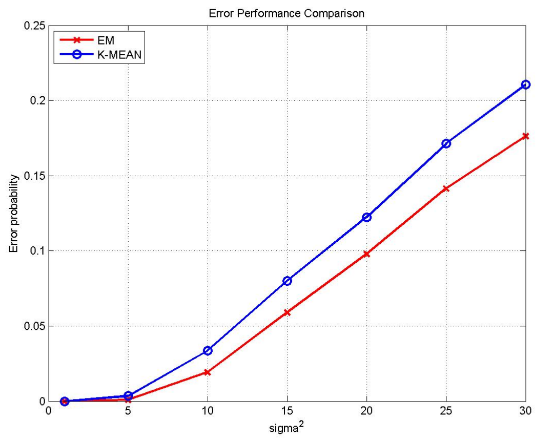
Typical plots are shown in Figure 4 and 5.

5. A few details about spectral clustering.

- (5 points) Suppose that $\{u_1, \ldots, u_k\}$ and $\{\hat{u}_1, \ldots, \hat{u}_k\}$ are any two orthonormal bases for the same k-dimensional null-space of a matrix L. Let U and $\hat{U}$ denote the $n \times k$ matrices whose columns are the respective orthonormal bases. Show that there is an orthonormal $k \times k$ matrix Q, for which $\hat{U} = UQ$. Converseley, show that if U is an $n \times k$ orthonormal matrix, and Q a $k \times k$ orthonormal matrix, then $\hat{U} = UQ$ is also orthonormal.
- (10 points) Recall that if we have a graph with k connected components, then the Laplacian has a k-dimensional subspace, spanned by k vectors, $\{u_1, \ldots, u_k\}$, where u_i has support only on the elements corresponding to the nodes in the i^{th} connected component. Hence, if we let U be the matrix with these vectors as *columns*, and then let $\{y_1, \ldots, y_n\}$ be the *rows* of U, then if we normalize each y_i, each point maps to the standard basis element $e_{c(i)}$, with $c(i)$ corresponding to the index of the connected component of i.

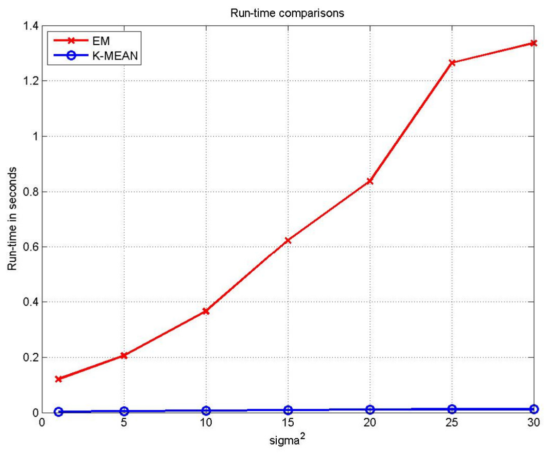
  Show that if instead of $\{u_1, \ldots, u_k\}$ we have any other orthonormal basis of the nullspace, $\{\hat{u}_1, \ldots, \hat{u}_k\}$, then if we form the matrix $\hat{U}$ in the same way, and let $\hat{y}_i$ be rows of $\hat{U}$, then again, $x_i = y_i/\|y_i\|$ will be one of k distinct, orthonormal vectors.


(a) Clusters obtained using EM

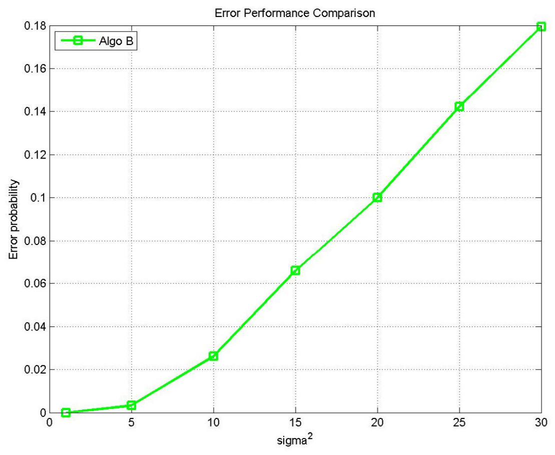


(b) Error performance of EM and K-Mean with increasing σ^2

Figure 4: EM vs K-Means for Gaussian mixture model



(a) Runtime comparison



(b) Error performance of Algorithm B

Figure 5: EM vs K-Means for Gaussian mixture model

Solution: $U = \{u_1, \ldots, u_k\}$, $\widehat{U} = \{\hat{u}_1, \ldots, \hat{u}_k\}$ are bases of the null space N of L. Hence both spans N. Hence each vector $\hat{u}_i$ can be expressed as linear combinations of the vectors $u_1, \ldots, u_k$. Let

$$\hat{u}_i = \sum_{j=1}^{k} q_{ji} u_j \quad \forall i = 1, \ldots, k$$

for some $q_{ji} \in \mathbb{R}$. In matrix notation we can simply write $\widehat{U} = UQ$. This proves existence of Q. Now since $\widehat{U}$ and U are orthonormal $\widehat{U}^T\widehat{U} = U^TU = I$. We can write

$$I = \widehat{U}^T\widehat{U} = Q^TU^TUQ = Q^TIQ = Q^TQ$$

Hence Q is an orthonormal matrix.

Conversely if U and Q are orthonormal matrix and $\widehat{U} = UQ$ then

$$\widehat{U}^T\widehat{U} = Q^TU^TUQ = Q^TIQ = Q^TQ = I$$

Hence $\widehat{U}$ is an orthonormal matrix.

Now for a graph with k connected components let U be the matrix,

$$U = [u_1 \ldots u_k] = \begin{bmatrix} y_1^T \\ \vdots \\ y_n^T \end{bmatrix}$$

where each u_j has support only on elements l when node l belongs to j^{th} cluster. U need not be orthonormal here since u_i's are any orthogonal vectors spanning the null space. Hence $u_j^Tu_j = \lambda_j > 0$, and $u_i^Tu_j = 0$ for $i \neq j$ as the clusters are non-overlapping. Clearly $\frac{y_j}{||y_j||} = e_{c(j)}$, when j^{th} node belongs to cluster $c(j)$.

For any other orthonormal matrix $\widehat{U} = [\hat{u}_1 \ldots \hat{u}_k]$ spanning the same null space it can be written as $U = \widehat{U}Q$ for some orthogonal matrix $Q = [q_1 \ldots q_k]$. Note that

$$U^TU = \Lambda = \begin{bmatrix} \lambda_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_k \end{bmatrix} = Q^T\widehat{U}^T\widehat{U}Q = Q^TIQ = Q^TQ$$

Hence,

$$Q^{-1} = \Lambda^{-1}Q^T = \begin{bmatrix} \lambda_1^{-1}q_1^T \\ \vdots \\ \lambda_k^{-1}q_k^T \end{bmatrix}$$

Now $\widehat{U} = UQ^{-1}$. Hence the j^{th} row of $\widehat{U}$ is simply $\hat{y}_j^T = y_j^TQ^{-1} = ||y_j||e_{c(j)}Q^{-1} = ||y_j||\lambda_{c(j)}^{-1}q_{c(j)}^T$.

Again if we normalize the rows of $\widehat{U}$, $x_i = \hat{y}_i/||\hat{y}_i||$, we see that for $i \neq j$,

$$x_i^T x_j = \frac{\hat{y}_i^T \hat{y}_j}{||\hat{y}_i||\ ||\hat{y}_j||} = \frac{||y_i||\ ||y_j||\lambda_{c(i)}^{-1}\lambda_{c(j)}^{-1} q_i^T q_j}{||\hat{y}_i||\ ||\hat{y}_j||} = 0$$

Hence x_i's also form k distinct orthonormal vectors.

6. (10 points) Recall the example we did in class via direct calculation:

$$A = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \end{pmatrix}, \qquad \Delta = \begin{pmatrix} 0 & \beta \\ \beta & 0 \end{pmatrix}.$$

Let E_0 denote the smallest eigenvalue of A and F_0 the smallest eigenvalue of $(A + \Delta)$. Use the sin-theta theorem to bound $d_p(E_0, F_0)$.

Solution: We have $||\Delta||_2 = \beta$. Minimum eigenvalue of A is 0. The eigenvalues of $A+\Delta$ are roots of the equation $\det(xI - (A+\Delta)) = 0$. Hence the maximum eigenvalue is $\lambda_1 = \frac{\epsilon}{2}\left(1 + \sqrt{1 + \frac{4\beta^2}{\epsilon^2}}\right)$. From sin-theta theorem to get a tight bound we can choose δ arbitrarily close to but less than λ_1, say $\delta = \lambda_1 - \nu$ ($\nu > 0$ small). Therefore $d_p(E_0, F_0)$ can be bounded by,

$$d_p(E_0, F_0) \leq \frac{||\Delta||_2}{\delta} = \frac{\beta}{\lambda_1 - \nu} \leq \frac{\beta}{\epsilon}$$

In order to get the exact bound we need to calculate the value of $\sin\Theta$. Let $h = \frac{\beta}{\epsilon}$, $\Delta = \sqrt{1 + 4h^2}$. We have,

$$\begin{aligned} e_0 &= (1, 0) \\ f_0 &= \frac{1}{\sqrt{2\Delta + 2\Delta^2}}(1+\Delta, -2h) \\ < e_o, f_0 > &= \frac{1+\Delta}{\sqrt{2\Delta + 2\Delta^2}} = 1 - \frac{\beta^2}{2\epsilon^2} + O(\beta^4) \end{aligned}$$

Therefore,

$$d_p(E_0, F_0) = ||\sin\Theta||_2 = \sqrt{1- < e_0, f_0 >^2} = \sqrt{\frac{\beta^2}{\epsilon^2} - O(h^4)} \leq \frac{\beta}{\epsilon}$$