[Report for MATLAB_Project_IJ]

20200548 장재형

1-(a)

1-(a) 번 문제는 $X=\begin{bmatrix}X_1\ 1\\ X_2\ 1\end{bmatrix}_{10000\times 785}$인 X와 $y=\begin{bmatrix}1\\ -1\end{bmatrix}_{10000\times 1}$인 y값을 이용하여 $a^*=\text{argmin}\|Xa-y\|_2^2$인 least squre problem의 해를 구하는 함수를 짜는 문제이다. 만든 함수를 이용하여 연습용 데이터인 DX를 사용하여 정수 0과 1을 구분하도록 하는 a값을 구하고 이를 이용하여 $TX01$에 대한 분류의 정확도를 계산하는 문제이다.

먼저 해당 least square problem의 해를 구하는 함수를 만들기 위해 $a^*=\text{argmin}\|Xa-y\|_2^2$인 a를 구하는 방법을 생각해 보자. $Ax=b$의 best approximation solution을 구하기 위해 $K_n=span\{b,Ab,Ab^2,\cdots,A^{n-1}b\}$인 Krylov subspace를 생각해보자. 우리는 $x\in K_n$인 $\min\|Ax-b\|$를 만족하는 x를 구하여야 한다. 이를 위하여 Arnoldi iteration을 사용한다. $q_1=b/\|b\|$으로 하고 $v=Aq_k$인 v에 대하여 $j=1$에서 k까지 $h_{jk}=q_j'v,\ v=v-h_{jk}q_j$를 시행하여 준다. $h_{k+1,k}=\|v\|$이고, $q_{k+1}=\frac{v}{h_{k+1,k}}$이다. 이러한 반복 시행에 의하면 $v_{k+1}=h_{k+1,k}q_{k+1}=Aq_k-\sum_{j=1}^{k}h_{jk}q_j$임으로 $Aq_k=\sum_{j=1}^{k+1}h_{jk}q_j$임을 알 수 있다. 그러면 $AQ_k=A[q_1|q_2|\cdots|q_k]=[q_1|q_2|\cdots|q_{k+1}]\begin{bmatrix}h_{11}&h_{12}&\cdots&h_{1k}\\ h_{21}&h_{22}&\cdots&h_{2k}\\ 0&h_{32}&\cdots&h_{3k}\\ \vdots&\vdots&\vdots&\vdots\\ 0&0&0&h_{k+1,k}\end{bmatrix}$로 나타낼 수 있음을 알 수 있다. $x\in K_n$이면 Q_k의 colum이 K_k의 basis를 이룸으로, $x\in Q_k$라 할 수 있고 이는 $x=Q_ky$로 나타낼 수 있다. 결국 우리는 $\min\|Ax-b\|=\min\|b-AQ_ky\|$인 y를 구하는 문제로 볼 수 있다. 이는 $\min\|b-AQ_ky\|=\min\|b-Q_{k+1}H_{k+1,k}y\|=\min\|Q_{k+1}'b-H_{k+1,k}y\|$이며 $Q_{k+1}'b=\|b\|[1|0|\cdots|0|0]'$임으로 $\min\|b-AQ_ky\|=\min\|\|b\|e_1-H_{k+1,k}y_k\|$임을 알 수 있다. 결국 $\min\|\|b\|e_1-H_{k+1,k}y_k\|$을 만족하는 y를 구하여 $Ax=b$의 best approximation solution인 x를 구하게 되는 것이다. 이를 사용한 매트랩 함수가 gmres함수이다. 이 함수를 사용하여 DX에서 index가 1인 $X1$과 index가 2인 $X2$를 사용하여 $X=\begin{bmatrix}X_1\ 1\\ X_2\ 1\end{bmatrix}_{10000\times 785}$ 행렬을 만들고, $y=\begin{bmatrix}1\\ -1\end{bmatrix}_{10000\times 1}$을 설정한 후 매트랩의 gmres함수는 정사각행렬을 사용함으로 noremal equation을 만들어 gmres(X'*X,X'*y)로 $a^*=\text{argmin}\|Xa-y\|_2^2$을 구하였다.

만든 함수로 구한 a'의 모습은 figure 1과 같다.

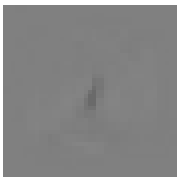

figure 1.a' image

$[x|1]a'$값이 구분의 정확도를 나타냄으로 가장 큰 값을 가지는 것이 연습용 데이터에 근거하여 가장 정확한 0의 모습일 것이고, 가장 작은 값을 가지는 것이 연습용 데이터에 근거하여 가장 정확한 1의 모습일 것임을 예측할 수 있다. $TX01$에서 가장 큰 값의 $[x|1]a'$값을 가지는 $TX01$의 이미지는 234번째 이미지이며 가장 작은 값의 $[x|1]a'$값을 가지는 $TX01$의 이미지는 903번째 이미지이다. 각각의 이미지는 figure 2와 같은 모습이다.

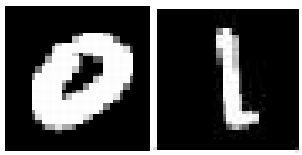

figure 2. 234th image(R) and 903th image(L)

x축을 $TX01$의 각 이미지 번호, y축을 각 이미지의 confidence로 한 그래프는 figure 3과 같다. $[x|1]a'$가 0보다 큰 값을 가지면 이 숫자는 0이고, 0보다 작은 값을 가지면 이 숫자는 1이다. 이를 이용해 $TX01$의 각 행이 어떠한 숫자를 나타내는지 구한 후 그 행에 해당하는 $TY01$값을 감하였을 때 0이 아닌 숫자가 나오게 되면 분류에 실패한 것이 된다. 이를 통해 479번째 $TX01$의 이미지가 분류에 실패했음으로 분류의 정확도는 $999/1000 = 0.999$ 즉, 99.9%라는 것을 알 수 있었다. figure 4는 분류에 실패한 $TX01$의 이미지이다.

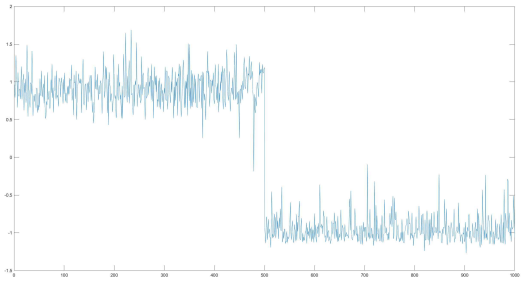

figure 3. confidence graph

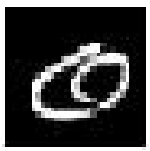

figure 4. 479th image

1-(b)

1-(b) 번 문제는 1-(a)번 문제에서 0과 1만을 구분하도록 했다면 이제는 정수$k-1$과 이외의 것들을 인식하도록 각각의 a_k'를 구하여 이를 그리고, TX에 대해 정확도를 계산하라는 문제이다.

먼저 각각의 a_k'를 구하는 방법에 대해 생각해 보자. 어떠한 정수$k-1$을 분류하는 a_k'는 1-(a)번 문제에서 사용하였던 least square problem의 해법과 같은 방식을 사용할 것이다. 단 1-(a)번 문제에서 사용한 $X1$을 정수 $k-1$에 해당하는 DX값을 대입하고 $X2$에는 정수 $k-1$이외에 해당하는 모든 DX값을 대입한다. 그 후 $y=\begin{bmatrix}1_{5000\times 1}\\-1_{45000\times 1}\end{bmatrix}$로 지정한다. 이렇게 하면 $[x\,|\,1]a_k'$은 어떠한 이미지 $[x\,|\,1]$가 특정한 정수$k-1$ 에 대해 얼마나 confidence를 가지는지 알 수 있다. 즉 $k=1,2,\cdots,10$가지의 confidence 값을 비교함으로써 이 숫자가 어떤 정수 $k-1$에 가장 가까운지를 알 수 있다.

위의 방식으로 구한 a_k'의 $k=1,2,\cdots,10$까지의 이미지는 figure 5와 같이 2×5 subplot에 도시하였다. 왼쪽 위의 이미지가 a_1'의 이미지이고, 오른쪽으로 갈수록 1식 커져 오른쪽 위의 이미지가 a_5'의 이미지, 왼쪽 아래가 a_6'의 이미지이다.

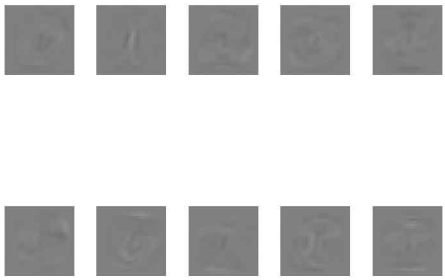

figure 5. image form a_1' to a_{10}'

이후 $(\arg\max[TX_i\,|\,1]a_k')-1$ (단, $k=1,2,\cdots,10$, $i=1,2,\cdots,10000$, TX_i는 TX행렬의 i행)을 각 행으로 하는 10000×1행렬을 구하면 우리가 구한 a_k'을 사용하여 TX의 이미지를 분류하여 각 이미지가 어떠한 정수를 나타내는지 알 수 있다. 이 10000×1행렬을 TY값과 빼 0이 되는 원소들의 개수만을 세어 보면 총 8680개의 이미지가 잘 분류되었음을 알 수 있다. 이를 통해 우리가 구한 a_k'를 사용하여 얻을 수 있는 분류의 정확도는 $8680/10000=0.8680$ 즉, 86.8%라는 것을 알 수 있다.

1-(c)

1-(c) 번 문제는 1-(a), 1-(b)번과 달리 least square problem의 해법을 사용하여 문제를 푸는 것이 아닌, PCA를 사용하여 문제를 푸는 것이다. PCA를 사용하면 principle component를 얻을 수 있는데 어떠한 정수 k의 DX에 대해 d개의 principle component를 얻어 이를 basis로 이용하는 subspace $span(W_k^d)$를 구한다. 이 subspace에 TX를 투영시킨 이미지의 노름을 분류의 confidence라 가정하는 것이다. $d = 1,2,\cdots,40$에 대해 각각의 정확도를 구하고, 1-(b)번보다 더 높은 가장 최소의 d값을 찾아보는 문제이다.

먼저 PCA는 어떠한 자료 $A = [x_1|x_2|\cdots|x_m]$에 대해 $u = \text{argmax}\sum_{i=1}^{m}|x_i'u|^2$인 u를 구하는 것이다. $\sum_{i=1}^{m}|x_i'u|^2 = u'AA'u$임으로 $\max\sum_{i=1}^{m}|x_i'u|^2 = \max u'AA'u$인 u를 찾는 문제와 같고, 이것을 만족하는 u는 AA'의 가장 큰 고윳값에 대한 고유벡터이다. matlab의 함수 pca는 A의 svd를 구한 후 left singular vector의 행렬을 도출한다. 이 문제를 해결하기 위해 특정 d값에 대해 XD에 근거한 PCA를 통해 TX를 분류한 R을 내놓는 함수를 구상했다. 이 함수는 먼저 DX의 0에서 9까지의 정수에 대해 $span(W^d)$를 구한 후 TX의 각 이미지에 대한 confidence를 도출하여 특정 행렬에 저장한다. 그 후 특정 행렬의 가장 큰 confidence 값을 가지는 열에 1을 뺀 값을 행으로 하는 R값을 산출하는 함수를 만들었다.

함수를 사용하여 각 정수에 대해 5개의 principle component를 그린 함수를 figure 6.의 10×5 subplot에 도시하였다. subplot의 m번째 행은 정수$m-1$의 principle component 5개를 보여준다.

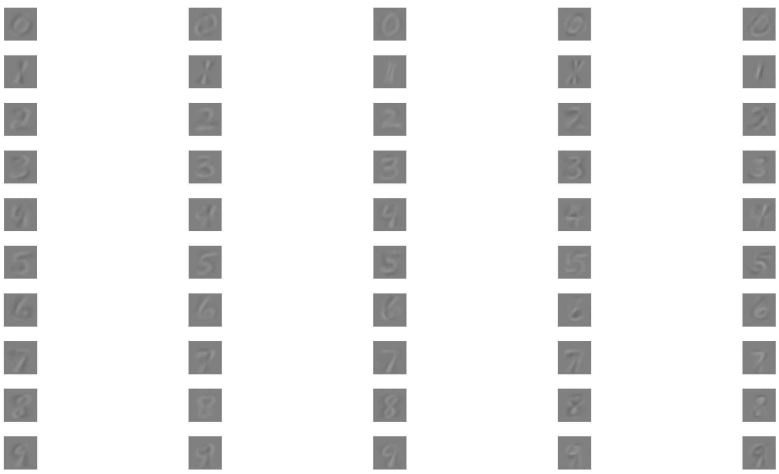

figure 6. image of each W_k^5 from k for 1 to 5

d값에 따른 TX에 대한 오류의 개수는 figure 7과 같다. figure 7의 x축은 d값으로 1에서 40까지의 정수이다. y축의 값은 분류의 정확도이며 주황색 그래프는 1-(b)의 정확도를, 파란색 그래프는 1-(c)의 정확도를 보여준다. 파란색 그래프를 보면, 초기 $d=1$일 때의 정확도는 36.14%였다가 마지막 $d=40$일 때의 정확도는 95.3%로 정확도가 현저히 높아짐을 볼 수 있다. 가장 작은 오류를 가지는 d의 값은 31로 453개 만의 오류를 가진다. 1-(b) 번 보다 더 높은 정확도를 가지는 최소의 d는 7으로 $d=7$이상부터 항상 1-(b) 번의 방법보다 더 높은 정확도를 보인다.

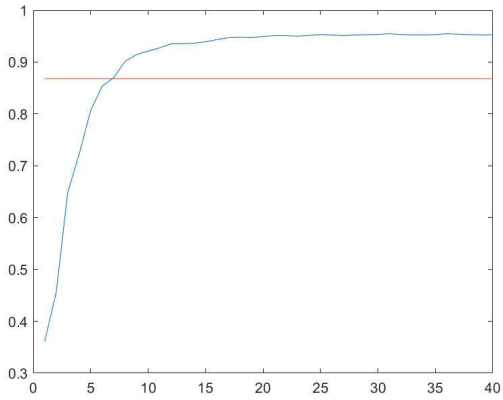

figure 7. accuracy of 1-(b) and 1-(a)

1-(b) 번과 1-(c) 번 방법의 차이점은 시도횟수의 차이다. 1-(b) 번에서는 특정 정수 $k-1$에 대해 하나의 $a_k^{'}$으로 least square problem을 해결하여 DX의 이미지 개수를 늘리는 것 이외에 더 높은 정확도를 얻는 방법이 없다. 하지만 1-(c) 번의 문제는 다양한 d값에 대해 PCA를 적용하여 노름을 구했기 때문에 특정 d이상의 값에 대해 같은 개수의 DX에도 불구하고 1-(b)보다 더 높은 정확도를 보여줄 수 있다. 이는 figure 5와 figure 6을 비교해 보아도 알 수 있는데, 각 figure 6이 5보다 어떠한 정수를 인식하는지 더 확연히 보여준다.

2.

2번 문제는 randomized linear algebra를 이용하여 200×1000의 크기가 큰 두 행렬의 곱을 uniform sampling과 norm-squared sampling을 사용하여 구하고, 각 sampling에 대한 예측값을 200회 반복한 후 평균과 분산, 상대적 오차를 구하여 이론과 맞는지 살펴보는 문제이다.

먼저, uniform sampling의 확률을 구하는 방법은 A행렬의 열의 개수의 역수로 k에 관계없이 일정하게 $p_k = 1/1000$의 값을 가진다. norm-squared sampling의 확률을 구하는 방법은 $p_k = \|A_k\|\|B_k'\| / \sum_{j=1}^{n} \|A_j\|\|B_j'\|$ 이다. 이것은 CR과 원래 곱의 오차 노름을 생각하여 노름을 최소화하는 방향으로 p_k값을 정한 것이다. 오차는 $E[\|CR - E[CR]\|_F^2]$ $= \sum_{k=1}^{n} \frac{\|A_k\|^2\|B_k'\|^2}{sp_k} - \frac{\|AB\|_F^2}{s}$ 이다. 뒤에 항은 상수항이므로 $k = 1,2,\cdots,n$의 p_k에 대해 앞의 항을 $\sum_{i=1}^{n} p_i = 1$임을 이용해 라그랑주 소거법을 사용하면 norm-squared sampling의 확률을 구할 수 있다. figure 8은 x축은 k값, y축은 확률로 파란색의 norm-squared sampling과 주황색의 uniform sampling의 확률을 보여준다.

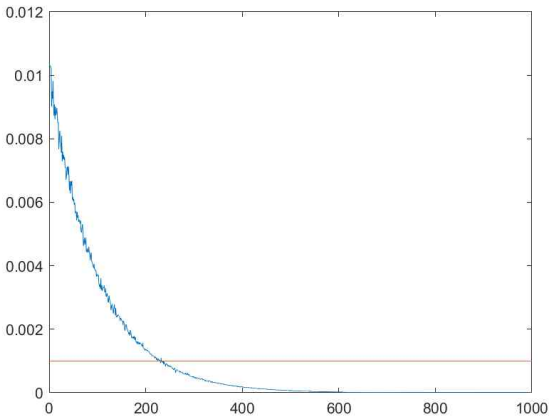

figure 8. uniform sampling and norm-squared sampling

본 과제에서 모든 sampling에 대해 사용한 각 k에 대한 히스토그램은 figure 9와 같다. x축은 사용된 k, y축은 사용된 k의 횟수를 나타낸다. $k(j)$는 난수 생성기를 사용하여 1부터 1000까지의 난수 중 20개를 선택하였고 이를 200번 반복할 것이기 때문에 행이 200개이고 각 행이 1부터 1000까지의 난수 중 20개인 200×10의 행렬이다.

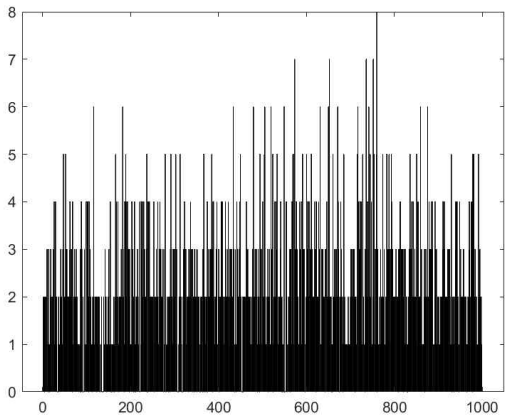

figure 9. histogram of number of sampling

문제에서 언급된 $AB \approx CR = \sum_{j=1}^{s} \frac{A_{k(j)} B_{k(j)}}{s p_{k(j)}}$를 사용하여 200개의 $k(j)$에 대한 200개의 CR을 구하였다. 먼저 200개의 uniform sampling에 의한 CR의 relative error 값은 figure 10과 같이 나타난다. x축은 N번째 CR이고 y축은 N번째 CR에 해당하는 relative error 값이다. 이를 이용해 error의 평균을 구한 후 error의 분산을 구하면 6.3349e-17의 분산이 나온다. 그 후 위에서 구한 norm-squared sampling에 의한 p_k에 대해 200개의 CR을 만들어 구한 relative error 값은 figure 11과 같다. 그래프의 개형은 figure 10과 같다. 이를 이용해 error의 평균을 구한 후 error의 분산을 구하면 1.2681e+16의 분산이 나온다.

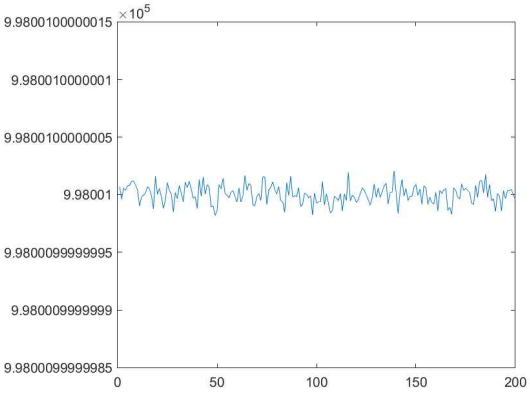

figure 10. uniform sampling error

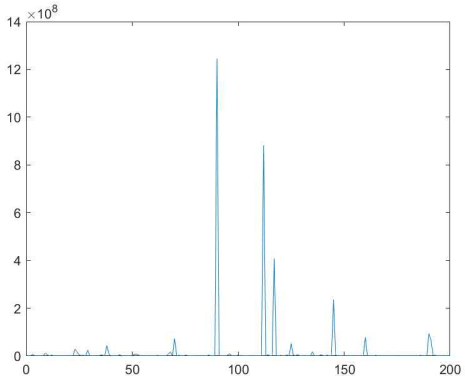

figure 11. norm-squared sampling error

randomized linear algebra를 이용하여 크기가 큰 두 행렬의 곱을 예측해 보았다. 이론상 norm-squared sampling을 사용한 오차의 분산 값이 uniform sampling을 사용한 오차의 분산 값보다 더 작게 나와야 함에도, 실제로 randomized linear algebra를 이용하여 구한 결과에서는 norm-squared sampling을 사용한 오차의 분산 값이 더 크게 나옴을 알 수 있었다. 하지만 figure 10과 figure 11의 그래프를 관찰하면 10의 그래프는 다소 일정하게 높은 오차율을 보이지만 11은 한 번씩 오차가 크게 튀는 성향을 보여준다. 이러한 몇몇 특수한 자룟값에 의해 norm-squared sampling을 사용한 오차의 분산이 더 크게 나온다고 생각하였다. 더하여 uniform sampling을 사용한 오차의 최솟값은 norm-squared sampling을 사용한 오차의 최솟값보다 훨씬 크게 나타나는 것 또한 관측할 수 있었다. 이를 통해 어느 정도 이론이 맞는다는 것을 유추할 수 있다.