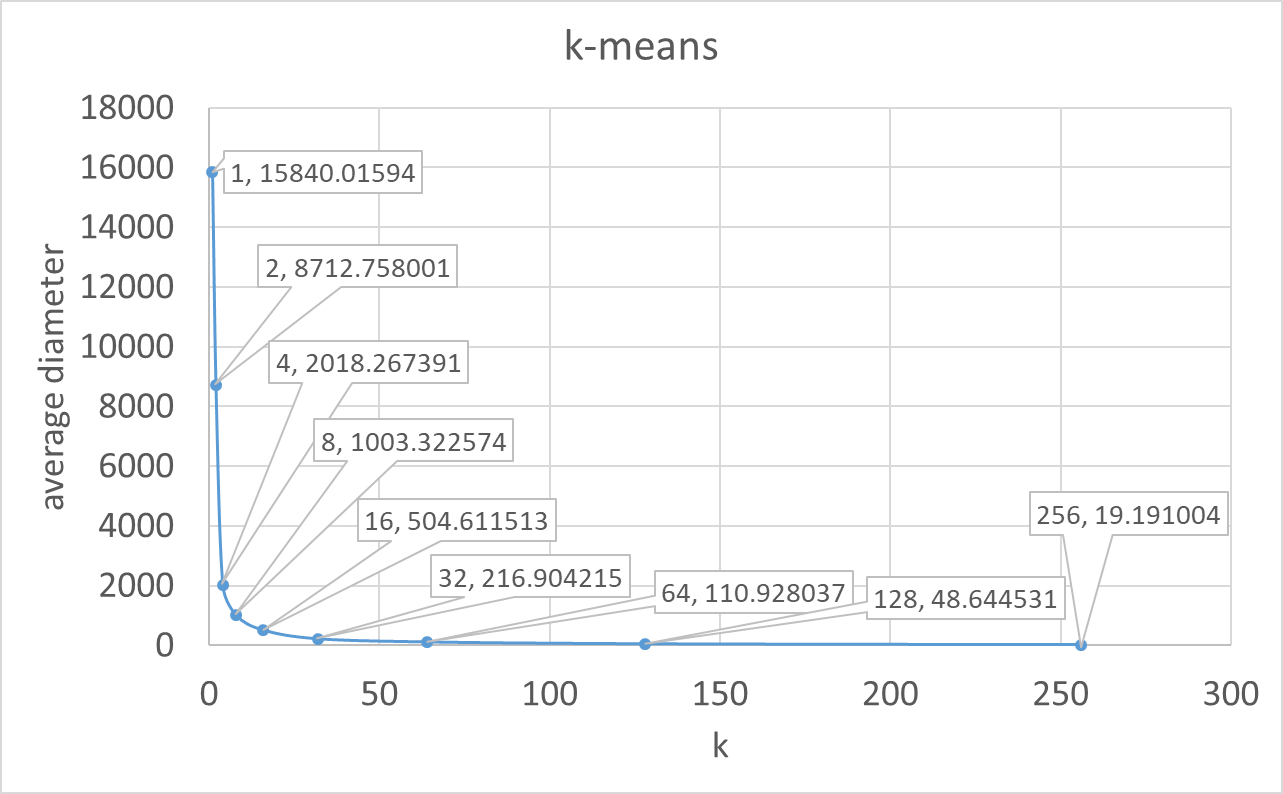
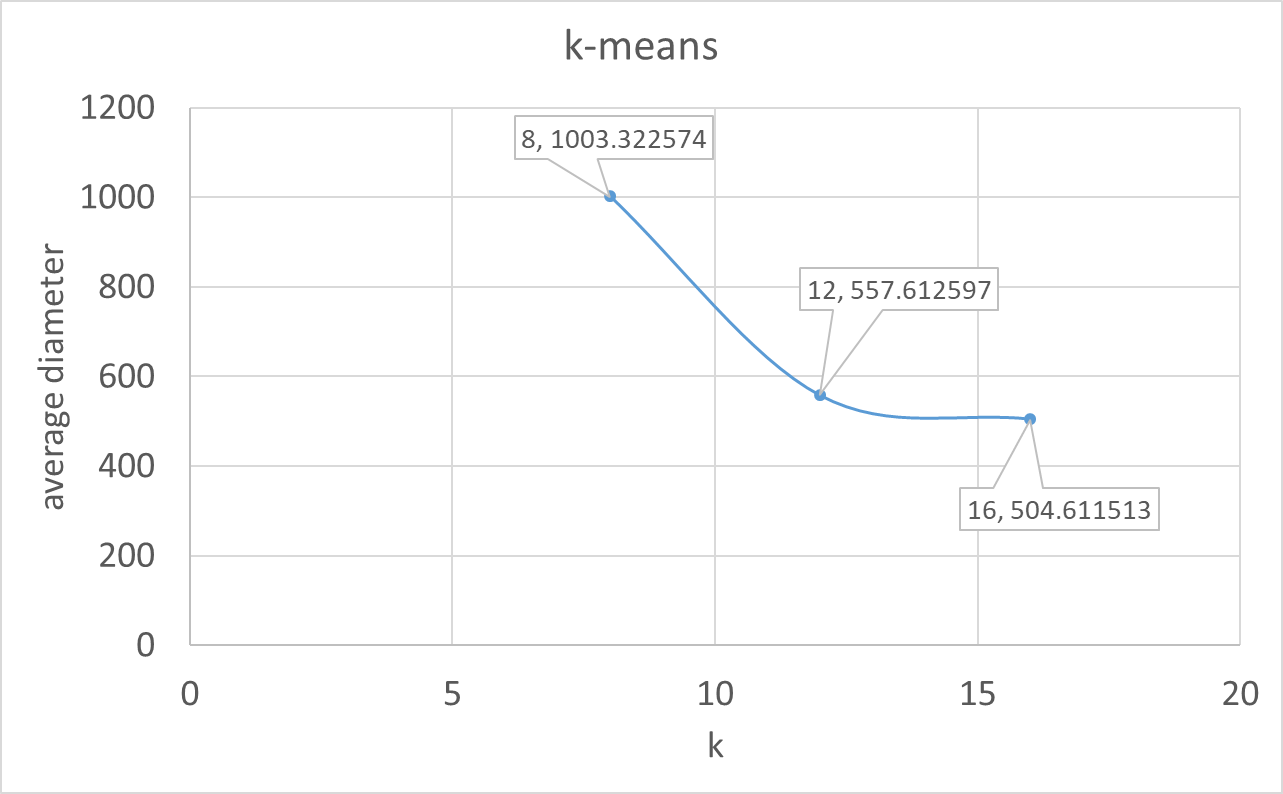
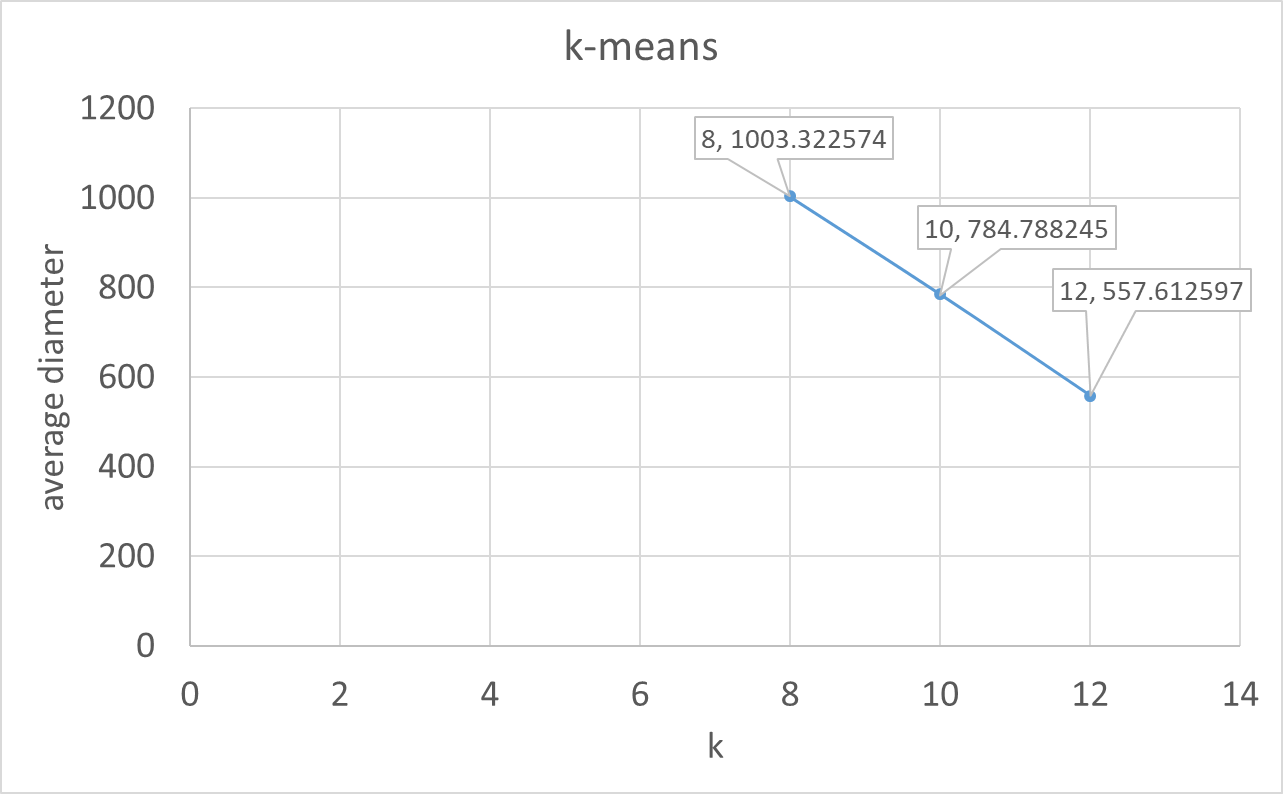
1-(b).



그래프를 보면 average diameter가 k값이 증가함에 따라 급격히 줄어들다가 k=16 ~ k=32 사이에서 거의 줄어들지 않게 되는 것을 볼 수 있다. 따라서 정확한 k 값은 k=8 ~ k=16 사이에 위치한다는 결론을 내릴 수 있다. 따라서 이진 탐색을 위해 중앙 값인 k=12일 때를 측정해보았다.



급격한 변화는 k=8에서 k=12 사이에 있기 때문에 다시 이진 탐색을 위해 k=10일때를 보았다.

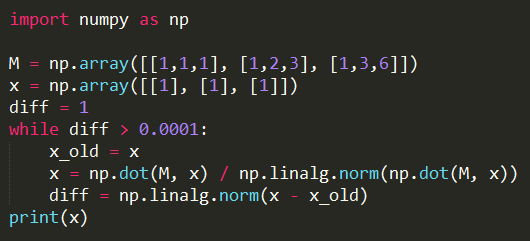


k=8 ~ k=12 사이에서는 diameter가 선형으로 감소하는 모습을 보였다. 급격히 감소하는 구간이 없기 때문에 이 데이터와 맞는 클러스터 개수는 선형 구간의 중점인 **10개**라고 할 수 있다.

2.

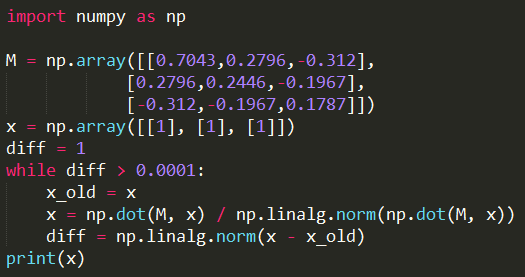
Exercise 11.1.7

(a).

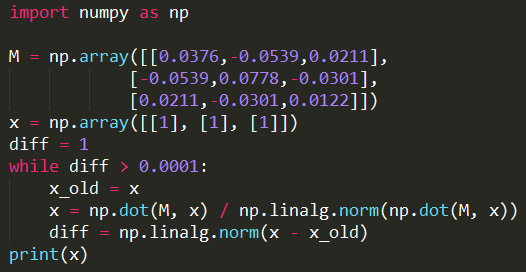


while문을 이용하여 power iteration을 구현하였다.

(d).

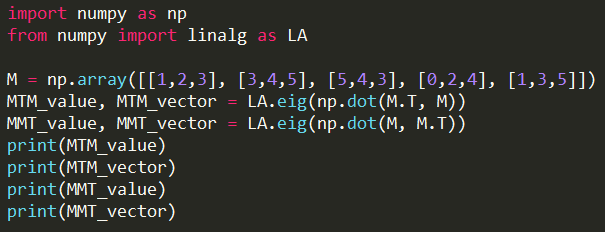


(e).



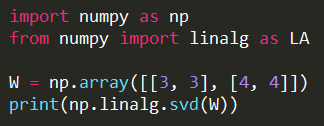
Exercise 11.3.1

(b) ~ (c)

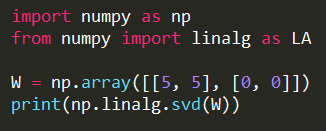


Exercise 11.4.2

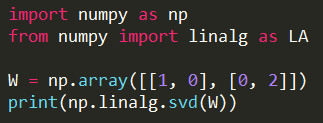
(a).



(b).



(c).



3-(c).

실행방법

Python hw2\_3c.py path/to/ratings.txt path/to/movies.txt path/to/ratings\_test.txt

알고리즘

이 문제에서는 1. Utility Matrix 정규화, 2. Matrix Factorization, 3. Gradient Descent, 4. Regularization, 5. Genre가 사용되었다. 먼저 3-(b)와 같이 Utility Matrix에서 각 User의 average rating을 뺐다. 여기서 뺐을 때 0이 되는 element는 0.00001로 두었는데, 이는 Gradient Descent 과정에 이 element가 포함되도록 하기 위함이었다.

그 후 k = 100으로 설정하여 U = [#\_user X 100], V = [100 X #\_movie] 행렬을 평균이 0이 되도록 초기화하였으며 이를 Gradient Descent 알고리즘을 통해 UV와 Utility Matrix M의 MSE가 최소가 되도록 했다. 여기서 MSE는 0이 아닌 element들에 대해서만 비교했으며, 따라서 0.00001로 설정한 것이다.

Gradient Descent 알고리즘은 Cost를 최소화하는 방향으로 각 U, V element들을 조정하였다. 여기서 Over-fitting을 피하기 위해 Regularization Term을 두었는데, 따라서 Cost를 MSE + (U, V의 각 element 제곱의 합) 로 설정하였다. 따라서 Learning Rule은 element = element – learning\_rate \* (d\_Cost / d\_element) 이며 여기서 Cost에는 MSE 뿐만 아니라 Regularization Term도 포함되어 있는 것이다.

이를 통해 학습이 모두 끝나면, M\* = UV가 되어 M\*를 참고하여 ratings\_test.txt에 있는 (user, movie) 쌍들에 대해 predict rating을 출력할 수 있다. 하지만 문제는 ratings.txt에 포함되어있지 않은 영화들이었는데, 이때 movies.txt에 있는 영화 장르를 이용했다.

User A에 대해 처음 보는 movie B에 대한 평점을 예측할 때, user A가 평가했던 영화들 중 movie B와 단 하나라도 공통된 장르가 있는 영화들을 골라 평점을 평균내었다. 이 평균이 movie B에 대한 예측 평점이 된다.