LATENT FACTOR MODELS

3.6.5 Singular Value Decomposition

- SVD는 matrix factorization의 형태입니다. matrix U, V는 mutually orthogonal 합니다.

  ** mutually orthogonal 하다는 것은 matrix의 한 col.이 다른 col 들에 완전히 독립적이게 됩니다. 이는 기하학적 해석을 용이하게 해줍니다.

- 하지만 이러한 decomposition의 Semantic interpretation는 일반적으로 더 어렵습니다. 왜냐하면 이러한 latent vectors는 pos와 neg quantities 모두를 가지기 때문입니다.
- *fully specified matrix* 에서 eigendecomposition methods를 이용하여 SVD를 수행하기 더 용이합니다.
- fully specified 된 ratings matrix를 고려해보겠습니다. 우리는 *truncated SVD* 를 이용하여 rating matrix R을 근사적으로 factorize할 수 있습니다.

$$R \approx Q_k \Sigma_k P_k^T \qquad k << min\{m, n\}$$

- 여기서 Q_k , Σ_k 그리고 P_k는 matrices로 사이즈는 각각 (m x k) , (k x k), (n x k) 입니다.
- Q_k 와 P_k 는 RR^T와 R^TR 에서 k 개의 가장 큰 eigenvectors입니다. Σ_k는 diagonal matrix로 (non-negative) square roots of the k lagest eigenvalues of either matrix along its diagonal 입니다.
- 여기서 한 가지 특징은 RR^T 와 R^TR 의 nonzero eigenvalues는 같다는 것입니다.
- P_k는 R^TR의 상위 k개의 eigenvectors를 갖고 있는데 이는 행공간에서 차원 축소를 위해 필요한 reduced basis representation입니다.
- 이러한 eigenvector는 ratings간의 item-item correlations의 방향에 대한 정보를 가지고 있습니다.
- 덕분에 줄어든 차원에서 각 유저를 표현하는 것이 가능해집니다.
- $Q_k\Sigma_k$ 는 transformed and reduced (m x k) representation of the original ratings matrix in the basis corresponding to P_k 입니다.
- SVD를 세 개의 matrices로 표현했는데 간단하게 두 개의 matirces로 표현 가능합니다.

$$U = Q_k\Sigma_k$$
$$V = P_k$$

- R = UV^T 로 정의 됩니다. user 와 item factor matrices가 orthogonal columns를 가진다면 쉽게 SVD로 분해하여 사용할 수 있습니다.(As long as the user and item factor matrices have orthogonal columns, it is easy to convert the resulting factorization into a form that is compliant with SVD)
- Factorization process의 목표는 orthogonal columns를 가지는 행렬 U, V를 찾는 것입니다.

$$MinmizeJ = \frac{1}{2}||R - UV^T||^2$$
$$subject\ to:$$
$$Columns\ U\ are\ mutually\ orthogonal$$
$$Columns\ V\ are\ mutually\ orthogonal$$

- 행렬의 col 들이 직교해야한다는 제약 조건이 있지만, 제약 조건이 없는 SVD와 J의 최적값이 동일하다고 합니다.

3.6.5.1 A Simple Iterative Approach to SVD

- R이 완전하지 않은 행렬인 경우 어떻게 최적화 하는지 알아보겠습니다.
- 첫 번째 스텝은 row 방향으로 평균을 구한 뒤 다 빼주어 mean-center R을 구합니다. 즉 유저별 평균 rating μ_i를 구한 뒤 row 방향으로 빼줍니다. 이때 이 평균 값은 저장해 두었다가 나중에 missing value에 대한 평점을 예측할때 다시 더 해주게 됩니다.
- 이렇게 구한 R_c 에서 missing value는 0으로 채워주면 됩니다.
- SVD를 사용해서 decomposition을 구합니다. $R_c = Q_k \Sigma_k P_k^T$
- i-th row of U는 k-dim vector 입니다. $\overline{u_i}$
- j-th row of V는 k-dim vector 입니다. $\overline{v_j}$
- $\hat{r_{ij}}$ 은 $\overline{u_i}$ 와 $\overline{v_i}$의 dot product로 구합니다.

$$\hat{r_{ij}} = \overline{u_i} \cdot \overline{v_i} + \mu_i$$

- 이 방법의 가장 큰 문제점은 missing entries를 row-wise means로 대체하는 것이 bias를 줄 수 있다는 것입니다.
- bias를 줄이는 방법으로는 여러가지가 있습니다. 한 가지 방법은 maximum-likelihood estimation입니다.
- 또 다른 방법은 missing entries에 대한 예측을 높여가면서 bias를 반복적으로 조금씩 줄여나가는 것입니다.
    1. Initialization : i-th row of R의 missing entries 를 μ_i로 초기화 합니다. => R_f 행렬
    2. Iterative step 1 : rank-k SVD of R_f를 수행합니다.
    3. Iterative stpe 2 : (originally) missing entries of R_f만 readjust 합니다. 그리고 다시 step 1으로 돌아갑니다.

- 처음에는 bias가 생기겠지만, 이후 iterations에서는 더 robust한 예측이 가능해집니다.
- 이런 접근방법은 missing entries가 클 경우 local optimum에 빠지게 됩니다. 또 convergence한 local optimum은 initialization에 따라 민감하게 바뀔 수 있습니다.
- Regularization은 앞에서 언급한 iterative method에 결합되어 사용될 수 있습니다. **관련 논문 링크**

3.6.5.2 An Optimization-Based Approach

- 앞에서 언급한 방법은 많은 연산량을 필요로 합니다. 왜냐하면 fully specified matrices를 사용하기 때문입니다. 대안으로 smaller matrices를 사용할 수도 있지만 이는 large-scale settings에서 좋은 성능을 기대하기 어렵습니다.
- 좀 더 효과적인 방법으로 앞에서의 optimization model에 orthogonality constraints를 추가하는 방법이 있습니다.

$$Minimize J = \frac{1}{2}\sum_{(i,j)\in S}(r_{ij} - \sum_{s=1}^{k} u_{is}\cdot v_{js})^2 + \frac{\lambda_1}{2}\sum_{i=1}^{m}\sum_{s=1}^{k} u_{is}^2 + \frac{\lambda_2}{2}\sum_{j=1}^{m}\sum_{s=1}^{k} v_{js}^2$$

$$subject\ to:$$

$$Columns\ of\ U\ are\ mutually\ orthogonal$$

$$Columns\ of\ V\ are\ mutually\ orthogonal$$

3.6.5.3 Out-of-Sample Recommendations

- 대부분의 matrix factorization과 같은 matrix completion methods는 *transduction* 합니다. 즉 학습 시점에 이미 ratings matrix에 포함된 users와 items에 대해서 예측을 할 수 있습니다.
- 즉 rating 행렬 R에 포함되어 있지 않은 새로운 users와 items에 대해서 예측을 하기 어려울 수 있습니다.
- orthogonal basis vectors의 한 가지 장점은 새로운 유저나 아이템의 out-of sample recommendations에서 좀 더 쉽게 좋은 성능을 뽑아 낼 수 있다는 것입니다. 이러한 문제를 inductive matrix completion이라고 합니다.

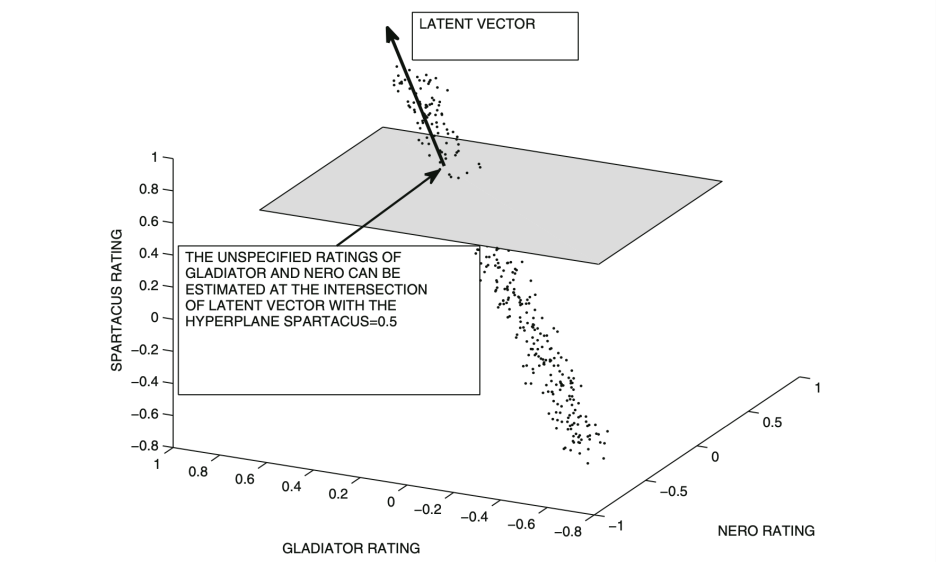

Figure 3.6: Leveraging correlation-based redundancies in missing data estimation for a user whose only specified rating is a value of 0.5 for the movie *Spartacus*

- figure 3.6를 통해 왜 orthogonal basis vectors 가 missing ratings의 예측에 도움이 되는지 이해해볼 수 있습니다.
- latent factors U와 V를 얻었다고 가정해봅시다. V의 columns는 원점을 지나는 k-dimensional hyperplane H_1 을 정의합니다.

> figure 3.6 공간 어디??

- figure 3.6에서 latent factors의 number는 1입니다. 그러므로 single latent vector(1-dim hyperplane)로 보이게 되는 것입니다. 2 factors였다면, plane으로 보였을 것입니다.
- 이제 새로운 user를 고려해보겠습니다. 이 새로운 유저는 U, V의 latent factors에 반영되어 있지 않습니다.
- 이 new user가 총 h 개의 items에 ratings했다고 가정해보겠습니다. ratings이 가능한 space는 (n-h)-dim hyperplane 입니다. 여기 h는 고정이겠죠. 이 hypterplane 을 H_2라고 하겠습니다.
- 우리의 목표는 H_2에서 rating이 될 한 point를 특정하는 것입니다. 그리고 그 점은 가능한 H_1에 가까워야 합니다.
    1. H_1 and H_2 do not intersect : H_2에서 H_1 과 가장 가까운 위치가 반환 됩니다. sum-of-squares optimization problem으로 풀 수 있습니다.
    2. H_1 and H_2 Intersect at a unique point : 한 점에서 만난 다면 그 점이 예측 rating이 됩니다.
    3. H_1 and H_2 intersect on an t-dim hyperplane, where $t \geq 1$: 대응 되는 유저들의 평점 평균이 리턴됩니다.

- 이렇게 Orthogonality는 기하학적 해석을 통해 뛰어난 insight를 제공해줍니다. 이렇게 out-of-recommendation을 가능하게 해주는 것이 그 장점 중 하나입니다.

3.6.5.4 Example of Singular Value Decomposition

Table 3.2: Illustration of the Bayes method with a binary ratings matrix

Item-Id ⇒ / User-Id ⇓	1	2	3	4	5	6
1	1	-1	1	-1	1	-1
2	1	1	?	-1	-1	-1
3	?	1	1	-1	-1	?
4	-1	-1	-1	1	1	1
5	-1	?	-1	1	1	1

- missing value를 row-wise 평균으로 채우겠습니다.

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -0.2 & -1 & -1 & -1 \\ 0 & 1 & 1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & 0.2 & -1 & 1 & 1 & 1 \end{pmatrix}$$

- rank-2 truncated SVD를 적용합니다.

$$R_f \approx \begin{pmatrix} 1.129 & -2.152 \\ 1.937 & 0.640 \\ 1.539 & 0.873 \\ -2.400 & -0.341 \\ -2.105 & 0.461 \end{pmatrix} \begin{pmatrix} 0.431 & 0.246 & 0.386 & -0.518 & -0.390 & -0.431 \\ -0.266 & 0.668 & -0.249 & 0.124 & -0.578 & 0.266 \end{pmatrix}$$

$$= \begin{pmatrix} 1.0592 & -1.1604 & 0.9716 & -0.8515 & 0.8040 & -1.0592 \\ 0.6636 & 0.9039 & 0.5881 & -0.9242 & -1.1244 & -0.6636 \\ 0.4300 & 0.9623 & 0.3764 & -0.6891 & -1.1045 & -0.4300 \\ -0.9425 & -0.8181 & -0.8412 & 1.2010 & 1.1320 & 0.9425 \\ -1.0290 & -0.2095 & -0.9270 & 1.1475 & 0.5535 & 1.0290 \end{pmatrix}$$

- 한 번의 SVD로도 꽤 괜찮은 행렬을 얻을 수 있습니다. 물론 이 값들은 처음에 missing value를 row averages로 채우면서 bias가 들어가 있는 상태입니다. 원래 행렬 R의 missing value들에 대해서 SVD로 분해 하였다가 다시 복원한 값들로 채워 줍니다.

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 0.5581 & -1 & -1 & -1 \\ 0.43 & 1 & 1 & -1 & -1 & -0.43 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -0.2095 & -1 & 1 & 1 & 1 \end{pmatrix}$$

- 이렇게 업데이트된 행렬로 다시 SVD를 수행합니다. 여전히 bias가 끼어 있지만 그래도 이전보다 더 좋은 값들을 가집니다. 다시 한번 더 SVD를 수행해보겠습니다.

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 0.9274 & -1 & -1 & -1 \\ 0.6694 & 1 & 1 & -1 & -1 & -0.6694 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -0.5088 & -1 & 1 & 1 & 1 \end{pmatrix}$$

- 첫 SVD 보다 업데이트 되는 값의 폭이 작습니다. 한 번 더 SVD를 수행해보겠습니다.

$$R_f = \begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 0.9373 & -1 & -1 & -1 \\ 0.7993 & 1 & 1 & -1 & -1 & -0.7993 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -0.6994 & -1 & 1 & 1 & 1 \end{pmatrix}$$

- 역시 처음 보다 변화 폭이 더 작아 졌습니다. 제대로 iteration을 돈다면 점점 값의 변화는 작아지며 수렴하게 될 겁니다. 그리고 그렇게 많은 반복이 필요하지도 않습니다. 5-10회 정도면 충분하다고 합니다.
- 이 방법이 global optimum을 보장하진 않습니다. 특히 잘 못 된 초기화는 더 나쁜 결과를 낼 겁니다.
- 따라서 첫 missing value를 결정하기 위해 neighborhood model 같은 simple heuristic method를 사용하는 것이 좋을 수도 있습니다. 초기에 값을 잘 설정해야 빠르게 수렴할 수 있습니다.

3.6.6 Non-negative Matrix Factorization