Kate Lassiter

Problem 1 (Probabilistic matrix factorization) – 35 points

In this problem, you will implement the MAP inference algorithm for the matrix completion problem discussed in class. As a reminder, for users $u \in \mathbb{R}^d$ and movies $v \in \mathbb{R}^d$, we have

$$u_i \sim N(0, \lambda^{-1} I), \quad i = 1, \ldots, N_1, \qquad v_j \sim N(0, \lambda^{-1} I), \quad j = 1, \ldots, N_2.$$

We are given an $N_1 \times N_2$ matrix M with missing values. Given the set $\Omega = \{(i, j) : M_{ij} \text{ is measured}\}$, for each $(i, j) \in \Omega$ we model $M_{ij} \sim N(u_i^T v_j, \sigma^2)$.

You should run your code on the user-movie ratings dataset provided on Courseworks. For your algorithm, set $\sigma^2 = 0.25$, $d = 10$ and $\lambda = 1$. Train the model on the larger training set for 100 iterations. For each user-movie pair in the test set, predict the rating using the relevant dot product. Note that the mean rating has been subtracted from the data and you should not round your prediction to the nearest integer.

Math Derivations

The joint likelihood of $p(M_o, U, V)$ can be factorized as follows:

$$p(M_o, U, V) = \underbrace{\Big[\prod_{(i,j)\in\Omega} p(M_{ij}|u_i, v_j)\Big]}_{\text{conditionally independent likelihood}} \times \underbrace{\Big[\prod_{i=1}^{N_1} p(u_i)\Big]\Big[\prod_{j=1}^{N_2} p(v_j)\Big]}_{\text{independent priors}}.$$

The MAP solution for U and V is the maximum of the log joint likelihood

$$U_{\text{MAP}}, V_{\text{MAP}} = \arg\max_{U,V} \sum_{(i,j)\in\Omega} \ln p(M_{ij}|u_i, v_j) + \sum_{i=1}^{N_1} \ln p(u_i) + \sum_{j=1}^{N_2} \ln p(v_j)$$

Calling the MAP objective function $\mathcal{L}$, we want to maximize

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2} \|M_{ij} - u_i^T v_j\|^2 - \sum_{i=1}^{N_1} \frac{\lambda}{2} \|u_i\|^2 - \sum_{j=1}^{N_2} \frac{\lambda}{2} \|v_j\|^2 + \text{constant}$$

The squared terms appear because all distributions are Gaussian.

To update each u_i and v_j, we take the derivative of $\mathcal{L}$ and set to zero.

$$\begin{aligned} \nabla_{u_i} \mathcal{L} &= \sum_{j\in\Omega_{u_i}} \frac{1}{\sigma^2} (M_{ij} - u_i^T v_j) v_j - \lambda u_i = 0 \\ \nabla_{v_j} \mathcal{L} &= \sum_{i\in\Omega_{v_j}} \frac{1}{\sigma^2} (M_{ij} - v_j^T u_i) u_i - \lambda v_i = 0 \end{aligned}$$

We can solve for each u_i and v_j individually (therefore EM isn't required),

$$\begin{aligned} u_i &= \left(\lambda\sigma^2 I + \textstyle\sum_{j\in\Omega_{u_i}} v_j v_j^T\right)^{-1} \left(\textstyle\sum_{j\in\Omega_{u_i}} M_{ij} v_j\right) \\ v_j &= \left(\lambda\sigma^2 I + \textstyle\sum_{i\in\Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\textstyle\sum_{i\in\Omega_{v_j}} M_{ij} u_i\right) \end{aligned}$$

However, we can't solve for all u_i and v_j at once to find the MAP solution.
Thus, as with K-means and the GMM, we use a coordinate ascent algorithm.

Kate Lassiter

Algorithm

MAP inference coordinate ascent algorithm

Input: An incomplete ratings matrix M, as indexed by the set Ω. Rank d.

Output: N_1 user locations, $u_i \in \mathbb{R}^d$, and N_2 object locations, $v_j \in \mathbb{R}^d$.

Initialize each v_j. For example, generate $v_j \sim N(0, \lambda^{-1}I)$.

for each iteration **do**

- **for** $i = 1, \ldots, N_1$ **update user location**

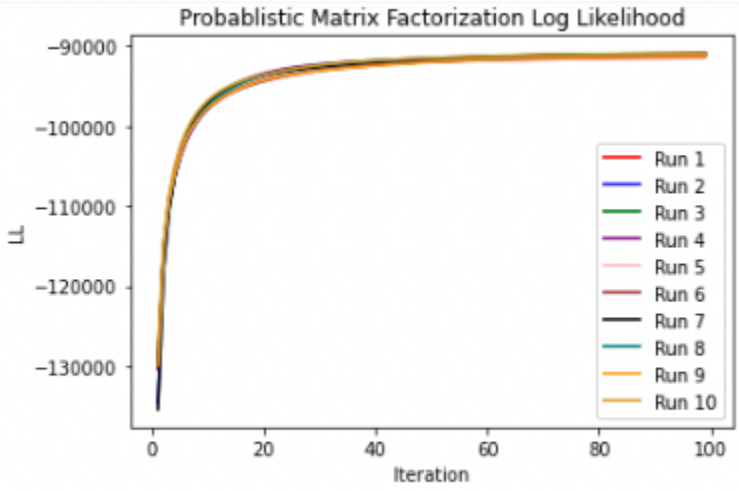
$$u_i = \left(\lambda\sigma^2 I + \sum_{j \in \Omega_{u_i}} v_j v_j^T\right)^{-1} \left(\sum_{j \in \Omega_{u_i}} M_{ij} v_j\right)$$

- **for** $j = 1, \ldots, N_2$ **update object location**

$$v_j = \left(\lambda\sigma^2 I + \sum_{i \in \Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\sum_{i \in \Omega_{v_j}} M_{ij} u_i\right)$$

Predict that user i rates object j as $u_i^T v_j$ rounded to closest rating option

a) Run your code 10 times. For each run, initialize your u_i and v_j vectors randomly using $N(0, I)$. On a *single* plot, show the the log joint likelihood for iterations 2 to 100 for each run. In a table, show in each row the final value of the training objective function next to the RMSE on the testing set. Sort these rows according to decreasing value of the objective function.



Final Training Objective	Test RMSE
-90912	1.09411
-90949	1.12533
-90955	1.10135
-90997.5	1.10852
-91008.4	1.10238
-91062.2	1.10269
-91113	1.13847
-91137.4	1.09506
-91294.3	1.11643
-91388.1	1.10417

b) For the run with the highest objective function value, pick the movies "Star Wars" "My Fair Lady" and "Goodfellas" and for each movie find the 10 closest movies according to Euclidean distance using their respective locations v_j. List the query movie, the ten nearest movies and their Euclidean distances. A mapping from index to movie is provided with the data.

Star Wars:

	Title	Distance
1	Empire Strikes Back, The (1980)	0.309821
2	Return of the Jedi (1983)	0.604593
3	Raiders of the Lost Ark (1981)	0.630327
4	My Man Godfrey (1936)	0.718611
5	Indiana Jones and the Last Crusade (1989)	0.863575
6	Star Trek: The Wrath of Khan (1982)	0.91754
7	Usual Suspects, The (1995)	0.942468
8	Terminator 2: Judgment Day (1991)	0.95176
9	Toy Story (1995)	0.986257
10	Searching for Bobby Fischer (1993)	0.988437

Goodfellas:

	Title	Distance
1	Godfather: Part II, The (1974)	0.724074
2	Full Metal Jacket (1987)	0.739328
3	Casino (1995)	0.781548
4	Once Upon a Time in the West (1969)	0.815049
5	Short Cuts (1993)	0.837442
6	Godfather, The (1972)	0.841819
7	Bonnie and Clyde (1967)	0.871418
8	Cool Hand Luke (1967)	0.883527
9	True Romance (1993)	0.891251
10	Pulp Fiction (1994)	0.916629

Kate Lassiter

My fair Lady:

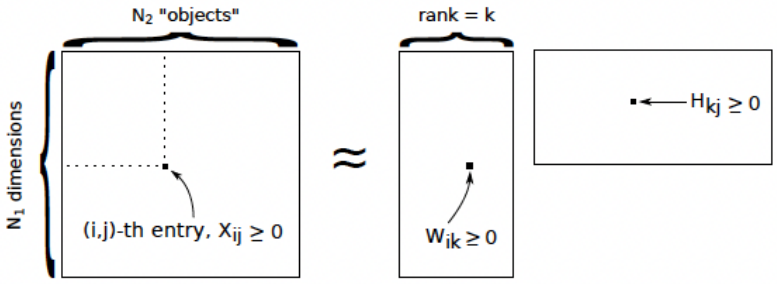
	Title	Distance
1	Sound of Music, The (1965)	0.721337
2	Singin' in the Rain (1952)	0.776565
3	Mary Poppins (1964)	0.810296
4	Victor/Victoria (1982)	0.825939
5	Shadowlands (1993)	0.82922
6	Sense and Sensibility (1995)	0.903853
7	American in Paris, An (1951)	0.953156
8	Emma (1996)	0.975409
9	Snow White and the Seven Dwarfs (1937)	1.01051
10	SubUrbia (1997)	1.02523

Problem 2 (Nonnegative matrix factorization) – 40 points

In this problem you will factorize an $N \times M$ matrix X into a rank-K approximation WH, where W is $N \times K$, H is $K \times M$ and all values in the matrices are nonnegative. Each value in W and H can be initialized randomly to a positive number, e.g., from a Uniform(1,2) distribution.

The data to be used for this problem consists of 8447 documents from *The New York Times*. (See below for how to process the data.) The vocabulary size is 3012 words. You will need to use this data to construct the matrix X, where X_{ij} is the number of times word i appears in document j. Therefore, X is 3012×8447 and most values in X will equal zero.

a) Implement and run the NMF algorithm on this data using the *divergence penalty*. Set the rank to 25 and run for 100 iterations. This corresponds to learning 25 topics. Plot the objective as a function of iteration.



Divergence objective

$$D(X\|WH) = -\sum_i \sum_j [X_{ij} \ln(WH)_{ij} - (WH)_{ij}]$$

Kate Lassiter

Problem

$$\min \sum_{ij} \left[X_{ij} \ln \frac{1}{(WH)_{ij}} + (WH)_{ij} \right] \quad \text{subject to} \quad W_{ik} \geq 0, \; H_{kj} \geq 0.$$

Algorithm

- Randomly initialize H and W with nonnegative values.
- Iterate the following, first for all values in H, then all in W:

$$H_{kj} \leftarrow H_{kj} \frac{\sum_i W_{ik} X_{ij} / (WH)_{ij}}{\sum_i W_{ik}},$$

$$W_{ik} \leftarrow W_{ik} \frac{\sum_j H_{kj} X_{ij} / (WH)_{ij}}{\sum_j H_{kj}},$$

until the change in $D(X \| WH)$ is "small."

Interpretation

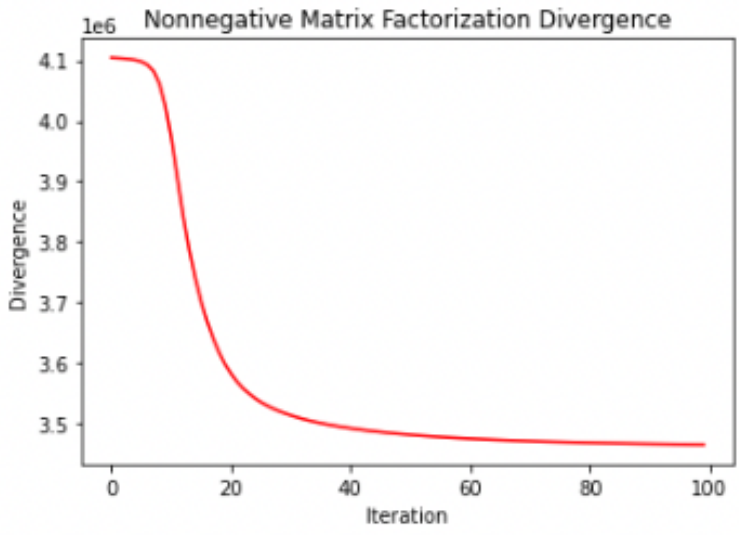
If we model the data as independent Poisson random variables

$$X_{ij} \sim \text{Pois}((WH)_{ij}), \qquad \text{Pois}(x|\lambda) = \frac{\lambda^x}{x!} e^{-\lambda}, \;\; x \in \{0, 1, 2, \dots\},$$

then the negative divergence penalty is maximum likelihood for W and H.

$$\begin{aligned} -D(X\|WH) &= \sum_{ij} [X_{ij} \ln (WH)_{ij} - (WH)_{ij}] \\ &= \sum_{ij} \ln P(X_{ij}|W,H) + \text{constant} \end{aligned}$$

We use: $P(X|W,H) = \prod_{ij} P(X_{ij}|W,H) = \prod_{ij} \text{Pois}(X_{ij}|(WH)_{ij})$.



Kate Lassiter

b) After running the algorithm, normalize the columns of W so they sum to one. For each column of W, list the 10 words having the largest weight and show the weight. The ith row of W corresponds to the ith word in the dictionary provided with the data. Organize these lists in a 5×5 table.

	Topic 1	Weight	Topic 2	Weight	Topic 3	Weight	Topic 4	Weight	Topic 5	Weight
1	music	0.019	doctor	0.018	win	0.023	sell	0.013	team	0.037
2	art	0.014	health	0.018	second	0.020	store	0.012	game	0.032
3	artist	0.012	care	0.017	hit	0.017	sale	0.011	player	0.026
4	dance	0.008	medical	0.015	third	0.016	president	0.010	play	0.024
5	song	0.008	hospital	0.015	victory	0.012	advertising	0.010	season	0.022
6	performance	0.008	patient	0.014	play	0.010	commercial	0.009	coach	0.014
7	play	0.007	treatment	0.011	finish	0.010	market	0.009	league	0.009
8	perform	0.007	receive	0.010	lose	0.009	industry	0.009	point	0.009
9	piece	0.007	drug	0.010	race	0.009	business	0.009	pass	0.008
10	exhibition	0.006	treat	0.010	score	0.009	big	0.009	football	0.007

	Topic 6	Weight	Topic 7	Weight	Topic 8	Weight	Topic 9	Weight	Topic 10	Weight
0	company	0.029	law	0.017	state	0.016	military	0.015	police	0.025
1	share	0.014	decision	0.014	vote	0.015	war	0.013	charge	0.022
2	financial	0.014	rule	0.014	campaign	0.014	government	0.012	case	0.014
3	executive	0.014	court	0.013	political	0.012	official	0.012	officer	0.014
4	stock	0.013	case	0.012	republican	0.012	american	0.011	crime	0.014
5	business	0.013	issue	0.010	election	0.011	states	0.010	trial	0.012
6	sell	0.011	legal	0.010	candidate	0.010	force	0.010	lawyer	0.012
7	chief	0.011	state	0.010	bill	0.010	attack	0.009	arrest	0.011
8	buy	0.010	lawyer	0.008	budget	0.010	leader	0.009	investigation	0.011
9	investment	0.010	government	0.008	party	0.010	peace	0.008	man	0.011

	Topic 11	Weight	Topic 12	Weight	Topic 13	Weight	Topic 14	Weight	Topic 15	Weight
0	editor	0.025	school	0.055	woman	0.026	company	0.035	study	0.012
1	article	0.017	student	0.042	child	0.021	computer	0.024	problem	0.011
2	write	0.011	mrs	0.028	man	0.019	information	0.018	expert	0.010
3	question	0.008	father	0.028	young	0.016	system	0.017	cause	0.009
4	writer	0.008	graduate	0.028	family	0.013	service	0.016	develop	0.008
5	issue	0.008	son	0.021	home	0.011	technology	0.015	research	0.008
6	point	0.007	college	0.018	life	0.011	site	0.011	require	0.007
7	study	0.006	daughter	0.018	wear	0.011	internet	0.011	risk	0.007
8	suggest	0.006	education	0.018	mother	0.010	customer	0.010	system	0.007
9	public	0.006	teacher	0.017	live	0.010	program	0.009	test	0.006

	Topic 16	Weight	Topic 17	Weight	Topic 18	Weight	Topic 19	Weight	Topic 20	Weight
0	play	0.020	member	0.017	food	0.012	thing	0.025	building	0.021
1	film	0.018	job	0.014	serve	0.008	really	0.014	city	0.020
2	movie	0.017	meeting	0.014	red	0.007	feel	0.013	house	0.013
3	character	0.015	group	0.012	restaurant	0.007	ask	0.013	area	0.013
4	television	0.013	official	0.011	water	0.007	lot	0.012	build	0.012
5	audience	0.011	union	0.011	small	0.007	put	0.011	resident	0.012
6	star	0.011	president	0.010	green	0.007	tell	0.010	street	0.010
7	production	0.010	city	0.010	white	0.007	happen	0.010	home	0.010
8	director	0.009	worker	0.009	dry	0.006	little	0.009	town	0.009
9	story	0.009	plan	0.009	eat	0.006	keep	0.009	open	0.009

	Topic 21	Weight	Topic 22	Weight	Topic 23	Weight	Topic 24	Weight	Topic 25	Weight
0	car	0.015	point	0.021	book	0.033	country	0.021	percent	0.045
1	mile	0.013	note	0.015	write	0.018	world	0.014	rate	0.018
2	hour	0.011	interest	0.014	life	0.016	american	0.014	increase	0.018
3	air	0.010	strong	0.012	story	0.011	power	0.012	rise	0.017
4	driver	0.009	market	0.011	die	0.010	states	0.010	average	0.014
5	travel	0.009	bond	0.010	read	0.010	political	0.009	price	0.014
6	train	0.009	issue	0.010	author	0.009	nation	0.009	low	0.012
7	fly	0.009	rate	0.009	newspaper	0.009	economic	0.007	growth	0.011
8	trip	0.008	economy	0.009	novel	0.009	national	0.007	fall	0.011
9	flight	0.008	move	0.009	name	0.009	decade	0.007	cost	0.010

Kate Lassiter

Recall

The divergence penalty is closely related mathematically to Latent Dirichlet Analysis (LDA).

Step 1. Form the term-frequency matrix X. ($X_{ij} = \#$ times word i in doc j)

Step 2. Run NMF to learn W and H using $D(X\|WH)$ penalty

Step 3. As an added step, after Step 2 is complete, for $k = 1, \ldots, K$

1. Set $a_k = \sum_i W_{ik}$
2. Divide W_{ik} by a_k for all i
3. Multiply H_{kj} by a_k for all j

Notice that this is does not change the matrix multiplication WH.

Interpretation: The kth *column* of W can be interpreted as the kth *topic*. The jth *column* of H can be interpreted as how much document j uses each topic.

Comments about Problem 2

- When dividing, you may get 0/0 = NaN. This will cause the algorithm to return all NaN values. You can add a very small number (e.g., 10^{-16}) to the denominator to avoid this.
- Each row in `nyt_data.txt` corresponds to a single document. It gives the index of words appearing in that document and the number of times they appear. It uses the format "idx:cnt" with commas separating each unique word in the document. Any index that doesn't appear in a row has a count of zero for that word. The vocabulary word corresponding to each index is given in the corresponding row of `nyt_vocab.dat`.