6.867
Problem Set 3

Dimitrios Koutentakis

November 2018

1 Kernel PCA and Locally linear Embedding

1.1

Here we show that $k_n(x_i, x_j) := ((I - W_n)^T(I - W_n))_{i,j}$ is a positive semi-definite kernel. That is we want to show that $k_n := ((I - W_n)^T(I - W_n))$ is a positive semi-definite matrix.

For $\mathcal{X} = \{x_1, ..., x_m\}$, we need:

$$X^T k X \geq 0$$
$$X^T (I - W_n)^T (I - W_n) X \geq 0 \quad (1)$$

If we let $(I - W_n) X = Z$, then equation 1 becomes:

$$Z^T \times Z = z_1^2 + z_2^2 + \cdots + z_m^2 \geq 0 \Rightarrow$$
$$X^T k X \geq 0.$$

1.2

Here we prove that the LLE kernel is positive semidefinite on $\{x_1, \cdots, x_m\}$. That is, we want to show that:

$$X^T \left((\lambda - 1)I + W_n^T + W_n - W_n^T W_n\right) X \geq 0$$
$$X^T(\lambda I - M)X \geq 0$$

However, we know from the finite-dimensional spectral theorem says that any Hermitian matrix can be diagonalized by a unitary matrix, and that the resulting diagonal matrix has only real entries. This implies that all eigenvalues of a Hermitian matrix A with dimension n are real, and that A has n linearly independent eigenvectors.

Thus can write X as $X = a_1 v_1 + a_2 v_2 \cdots a_n v_n$. Then, because $v_i^T v_j = 0 \forall i \neq j$, we get:

$$X^T \left((\lambda - 1)I + W_n^T + W_n - W_n^T W_n\right) X =$$
$$= \sum_i^m a_i^2 (\lambda_{max} - \lambda_i) v_i^2$$
$$\geq 0$$

because $\lambda_{max} \geq \lambda_i \forall i \in [1, m]$.

1.3

We know that the solutions of PCA have to be the eigenvalues corresponding to the largest eigenvalues of the matrix $\lambda^* I - M$. From those, we need to keep the ones that satisfy the given constraints. Let v_i be an eigenvector of M. Then:

$(\lambda^* I - M)v_i = \lambda^* v_i - M v_i = (\lambda^* - \lambda_i)v_i$, so the eigenvectors of $\lambda^* I - M$ are the eigenvectors of M, with the reverse order.

The two constraints that need to be satisfied are $\sum_{i=1}^m Y_i = 0$ and $\frac{1}{m}\sum_{i=1}^m Y_i Y_i^T = I$. The first constraint is equivalent to $\sum_{i=1}^m v_i = 0$ for any eigenvector v of M with non-zero eigenvalue. We will first show that the eigenvectors $\alpha^2, \alpha^3, \ldots, \alpha^{d+1}$ satisfy the first constraint. Note here that $\alpha_1 = e = (1, 1, \ldots, 1)^T$ is an eigenvector of M with eigenvalue 0, since $Me = (I - W)^T(I - W)e = (I - W)^T[(I - W)e] = 0$ using the fact that all rows of $I - W$ have sum 0.

Let $v^{(i)}$ be an eigenvector of M, $Z = I - W$ and Z_i be the columns of Z. Then, $\lambda_i v^{(i)} = M v^{(i)} = Z^T Z v^{(i)} \Rightarrow \lambda v_j^{(i)} = Z_j \sum_{k=1}^m Z_k^T v^{(i)} \Rightarrow \lambda_i \sum_{j=1}^m v_j^{(i)} = (\sum_{k=1}^m Z_k)(\sum_{j=1}^m Z_j v^{(i)}) = 0$, since $\sum_{k=1}^m Z_k = \sum_{k=1}^m (I - W)_k = 0$. Thus, for $\lambda_i \neq 0$ it is $\sum_{j=1}^m v_j^{(i)} = 0$. This means that the first constraint is satisfied if we use the set $\{\alpha^2, \alpha^3, \ldots, \alpha^{d+1}\}$ ($\alpha^1 = e$ is excluded because its eigenvalue is 0 for M).

Next, we need to show that the second constraint is also satisfied if we use the set $\{\alpha^2, \alpha^3, \ldots, \alpha^{d+1}\}$. Let $A = (\alpha^2 \quad \alpha^3 \quad \cdots \quad \alpha^{d+1})$ and define A_i to

be the i-th row of A. We want to show that $\frac{1}{m}\sum_{i=1}^{m} A_i A_i^T = I$ (the eigenvectors are up to the appropriate scalars). Let $N = \frac{1}{m}\sum_{i=1}^{m} A_i A_i^T$. Then: $N_{kl} = \sum_{i=1}^{m} A_{ik}A_{il} = \sum_{i=1}^{m} \alpha_i^{(k)}\alpha_i^{(l)} = \alpha^{(k)} \cdot \alpha^{(l)} = [[k = l]]$, since eigenvectors are orthogonal. Thus, $N = I$.

Since both constraints are satisfied for the set $S = \{\alpha^2, \alpha^3, \ldots, \alpha^{d+1}\}$ and α^1 is not valid, the LLE embeddings are the set S.

1.4

Let $N = (I - 1_m)(\lambda^* I - M)(I - 1_m) = (I - \frac{1}{m}ee^T)(\lambda^* I - M)(I - \frac{1}{m}ee^T)$. Then, $Ne = (I - \frac{1}{m}ee^T)(\lambda^* I - M)[(I - \frac{1}{m}ee^T)e] = 0$, since the rows of $I - \frac{1}{m}ee^T$ have sum 0. Also, let v be any other eigenvector of M. We proved before that $e^T v = 0$, so $Nv = (I - \frac{1}{m}ee^T)(\lambda^* I - M)(I - \frac{1}{m}ee^T)v = (\lambda^* I - M)v = (\lambda^* - \lambda)v$, so v is also an eigenvector of N. Thus all eigenvectors of N are the eigenvectors of M.

Contrary to part 1.3, the eigenvector e has eigenvalue 0 (before it had λ^*), so it is the last eigenvector of N. The set $\{\alpha^1, \alpha^2, \ldots, \alpha^d\}$ is then the same as the set S from part 1.3. This means that the constraints are satisfied.

1.5

We can think of the LLE kernel as a similarity measure of the coefficients of the two x's. We have: $k_{ij} = \{\lambda I - M\}_{ij} = -M_{ij} = -\{(I - W_n)^T(I - W_n)\}_{ij} = -\langle e_i - W_{n,i}, e_j - W_{n,j}\rangle$, where $e_i - W_{n,i}$ is the negative of the contributions of other x's in x_i. Then, the dot product keeps only the values which are non-zero in both x_i and x_j, so it keeps the common neighbors of the two. A larger product means more common neighbors, so a larger similarity between the two.

2 Variational Inference for LDA

2.1

$$\begin{aligned}\log p(\mathbf{w}|\alpha,\beta &= \log \int_\theta \sum_{\mathbf{z}} p(\mathbf{z},\theta,\mathbf{w}|\alpha,\beta)d\theta \\ &= \log \mathbb{E}_q[\frac{p(\mathbf{z},\theta,\mathbf{w}|\alpha,\beta)}{q(\mathbf{z},\theta)}] \\ &\geq \mathbb{E}_q[\log \frac{p(\mathbf{z},\theta,\mathbf{w}|\alpha,\beta)}{q(\mathbf{z},\theta)}] \\ &= \boxed{\mathbb{E}_q[\log p(\mathbf{z},\theta,\mathbf{w}|\alpha,\beta)] - \mathbb{E}_q[\log q(\mathbf{z},\theta)]}\end{aligned}$$

where for the above we used Jensen's inequality.

2.2

$$\begin{aligned}p(\mathbf{w},\mathbf{z},\theta|\alpha,\beta) &= p(\mathbf{w}|\mathbf{z},\theta,\alpha)p(\mathbf{z}|\theta,\alpha,\beta)p(\theta|\alpha,\beta) \\ &= p(\mathbf{w}|\mathbf{z},\beta)p(\mathbf{z}|\theta)p(\theta|\alpha) \Rightarrow\end{aligned}$$

$$\begin{aligned}\mathcal{L} =& \mathbb{E}_q[\log p(\mathbf{z},\theta,\mathbf{w}|\alpha,\beta)] - \mathbb{E}_q[\log q(\mathbf{z},\theta)] \\ =& \mathbb{E}_q[\log p(\mathbf{w}|\mathbf{z},\alpha,\beta)] + \mathbb{E}_q[\log p(\mathbf{z}|\theta)] + \\ & \mathbb{E}_q[\log p(\theta|\alpha)] - \mathbb{E}_q[\log q(\mathbf{z},\theta)] \qquad (2)\end{aligned}$$

Next, we know that $q(\theta,\mathbf{z}|\gamma,\phi) = q(\theta|\gamma)\prod_{n=1}^{N} q(z_n|\phi_n)$ and γ and ϕ are given in $\mathcal{L}$ as a parameter. Thus, we can simplify every term in Equation (2):

First term:

$$\begin{aligned}\mathbb{E}_q(\log p(\mathbf{w}|\mathbf{z},\beta)) &= \int_\theta \sum_{\mathbf{z}} q(\mathbf{z},\theta|\gamma,\phi)\log p(\mathbf{w}|\mathbf{z},\beta)d\theta \\ &= \int_\theta \sum_{\mathbf{z}} q(\theta|\gamma)\prod_{n=1}^{N} q(z_n|\phi_n)\log p(\mathbf{w}|\mathbf{z},\beta)d\theta \\ &= \sum_{\mathbf{z}}\prod_{n=1}^{N} q(z_n|\phi_n)\log p(\mathbf{w}|\mathbf{z},\beta)\int_\theta q(\theta|\gamma)d\theta \\ &= \sum_{\mathbf{z}}\prod_{n=1}^{N} q(z_n|\phi_n)\log p(\mathbf{w}|\mathbf{z},\beta) \\ &= \sum_{\mathbf{z}}\prod_{n=1}^{N} q(z_n|\phi_n)\sum_{n=1}^{N}\log p(w_n|z_n,\beta) \\ &= \sum_{\mathbf{z}}\prod_{n=1}^{N} q(z_n|\phi_n)\sum_{t=1}^{N}\log \beta_{z_t,w_t} \\ &= \sum_{\mathbf{z}}\sum_{t=1}^{N}(\log \beta_{w_t,z_t})\prod_{n=1}^{N} q(z_n|\phi_n) \\ &= \sum_{t=1}^{N}(\sum_{z_t} q(z_t|\phi_t)\log \beta_{z_t,phi_t}) \\ &\quad \times \sum_{\mathbf{z}\setminus z_t}\prod_{n=1,n\neq t}^{N} q(z_n|\phi_n) \\ &= \sum_{n=1}^{N}\sum_{z_n} q(z_n|\phi_n)\log \beta_{z_n,w_n} \\ &= \sum_{n=1}^{N}\sum_{z_n}\phi_{n,z_n}\log \beta_{z_n,w_n} \\ &= \boxed{\sum_{n=1}^{N}\sum_{i=1}^{k}\phi_{n,i}\log \beta_{i,w_n}}\end{aligned}$$

since $\int_\theta q(\theta|\gamma)d\theta = 1$ and $\sum_{\mathbf{z}\setminus z_t}\prod_{n=1,n\neq t}^{N} q(z_n|\phi_n) = 1$, by total probability theorem.

Second term:

$$\begin{aligned}\mathbb{E}_q[\log p(\mathbf{z}|\theta)] &= \int_\theta \sum_{\mathbf{z}} q(\mathbf{z},\theta|\gamma,\phi)\log p(\mathbf{z}|\theta)d\theta \\ &= \int_\theta \sum_{\mathbf{z}} q(\theta|\gamma)\prod_{n=1}^{N} q(z_n|\phi_n)\log p(\mathbf{z}|\theta)d\theta \\ &= \int_\theta \sum_{\mathbf{z}} q(\theta|\gamma)\prod_{n=1}^{N} q(z_n|\phi_n)\sum_{t=1}^{N}\log p(z_t|\theta)d\theta \\ &= \int_\theta q(\theta|\gamma)\sum_{t=1}^{N}\sum_{\mathbf{z}}\log p(z_t|\theta)\prod_{n=1}^{N} q(z_n|\phi_n)d\theta \\ &= \int_\theta q(\theta|\gamma)\sum_{t=1}^{N}\sum_{z_t}\log p(z_t|\theta)q(z_t|\phi_t)\end{aligned}$$

$$\times \sum_{\mathbf{z}\setminus z_t} \prod_{n=1, n\neq t}^{N} q(z_n|\phi_n) d\theta$$
$$= \int_\theta q(\theta|\gamma) \sum_{n=1}^{N} \sum_{z_n} \log p(z_n|\theta) q(z_n|\phi_n) d\theta$$
$$= \int_\theta q(\theta|\gamma) \sum_{n=1}^{N} \sum_{z_n} (\log \theta_{z_n}) \phi_{n,z_n}$$
$$= \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \int_\theta q(\theta|\gamma) \log \theta_i d\theta$$
$$= \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \mathbb{E}_q[\log \theta_i|\gamma]$$
$$= \boxed{\sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} (\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_i))}$$

Third term:

$$\mathbb{E}_q[\log p(\theta|\alpha)] = \int_\theta \sum_{\mathbf{z}} q(\mathbf{z}, \theta) \log p(\theta|\alpha) d\theta$$
$$= \int_\theta \sum_{\mathbf{z}} q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \log p(\theta|\alpha) d\theta$$
$$= \int_\theta q(\theta|\gamma) \log p(\theta|\alpha) d\theta \sum_{\mathbf{z}} \prod_{n=1}^{N} q(z_n|\phi_n)$$
$$= \int_\theta q(\theta|\gamma) \log p(\theta|\alpha) d\theta$$
$$= \int_\theta q(\theta|\gamma) (\log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1) \log \theta_i) d\theta$$
$$= \log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1) \int_\theta q(\theta|\gamma) \log \theta_i d\theta$$
$$= \log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1) \mathbb{E}_q[\log \theta_i|\gamma]$$
$$= \log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}) \gamma_j)$$

Fourth term:

$$\mathbb{E}_q[\log q(\mathbf{z}, \theta)] = \mathbb{E}_q[\log q(\theta|\gamma)] + \sum_{n=1}^{N} \mathbb{E}_q[\log q(z_n|\phi_n)]$$
$$\mathbb{E}_q[\log q(\theta|\gamma)] = \log \Gamma(\sum_{i=1}^{k} \gamma_i) - \sum_{i=1}^{k} \log \Gamma(\gamma_i) + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j))$$

$$\sum_{n=1}^{N} \mathbb{E}_q[\log q(z_n|\phi_n)] = \sum_{n=1}^{N} \int_\theta \sum_{\mathbf{z}} q(\mathbf{z}, \theta) \log q(z_n|\phi_n) d\theta$$
$$= \sum_{n=1}^{N} \int_\theta \sum_{\mathbf{z}} q(\theta|\gamma) \prod_{t=1}^{N} q(z_t|\phi_t) \log q(z_n|\phi_n) d\theta$$
$$= \sum_{n=1}^{N} \sum_{\mathbf{z}} \prod_{t=1}^{N} q(z_t|\phi_t) \log q(z_n|\phi_n) \int_\theta q(\theta|\gamma) d\theta$$
$$= \sum_{n=1}^{N} \sum_{\mathbf{z}} \prod_{t=1}^{N} q(z_t|\phi_t) \log q(z_n|\phi_n)$$
$$= \sum_{n=1}^{N} \sum_{z_n} q(z_n|\phi_n) \log q(z_n|\phi_n) \times \sum_{\mathbf{z}\setminus z_n} \prod_{t=1, t\neq n}^{N} q(z_t|\phi_t)$$
$$= \sum_{n=1}^{N} \sum_{z_n} q(z_n|\phi_n) \log q(z_n|\phi_n)$$
$$= \sum_{n=1}^{N} \sum_{z_n} \phi_{n,z_n} \log \phi_{n,z_n}$$
$$= \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \log \phi_{n,i}$$

$$\mathbb{E}_q[\log q(\mathbf{z}, \theta)] =$$
$$= \log \Gamma(\sum_{i=1}^{k} \gamma_i) - \sum_{i=1}^{k} \log \Gamma(\gamma_i) + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j)) + \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \log \phi_{n,i}$$

Thus:

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \log \beta_{i,w_n} + \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} (\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_i)) + \log \Gamma(\sum_{i=1}^{k} \alpha_i) - \sum_{i=1}^{k} \log \Gamma(\alpha_i) + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k}) \gamma_j) + \log \Gamma(\sum_{i=1}^{k} \gamma_i) - \sum_{i=1}^{k} \log \Gamma(\gamma_i) + \sum_{i=1}^{k} (\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j)) + \sum_{n=1}^{N} \sum_{i=1}^{k} \phi_{n,i} \log \phi_{n,i}$$

2.3

We will use Lagrange Multipliers: $L(\phi, \gamma) = \mathcal{L} - \sum_{n=1}^{N} \lambda_n (\sum_{i=1}^{k} \phi_{ni} - 1)$. By first order conditions:

$$\frac{\partial L}{\partial \phi_{ni}} = \log \beta_{iu} + \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) - \log \phi_{ni} - 1 - \lambda_n = 0$$

$$\Rightarrow \log \phi_{ni} = \log \beta_{iu} + \Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) - 1 - \lambda_n$$

$$\Rightarrow \phi_{ni} = \beta_{iu} \exp(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j) - 1 - \lambda_n)$$

$$\Rightarrow \boxed{\phi_{ni} \propto \beta_{iu} \exp(\Psi(\gamma_i) - \Psi(\sum_{j=1}^{k} \gamma_j))}$$

2.4

Again:

$$\begin{aligned}
\frac{\partial L}{\partial \gamma_i} =& \sum_{n=1}^{N} \phi_{ni} \Psi'(\gamma_i) - \sum_{n=1}^{N} \phi_{ni} \Psi'(\sum_{j=1}^{k} \gamma_j) \\
&+ (\alpha_i - 1)(\Psi'(\gamma_i) - \Psi'(\sum_{j=1}^{k} \gamma_j)) \\
&- \cancel{\Psi(\sum_{j=1}^{k} \gamma_j)} + \cancel{\Psi(\gamma_i)} - (\cancel{\Psi(\gamma_i)} - \cancel{\Psi(\sum_{j=1}^{k} \gamma_j)}) \\
&- (\gamma_i - 1)(\Psi'(\gamma_i) - \Psi'(\sum_{j=1}^{k} \gamma_j)) = 0 \Rightarrow \\
&(\gamma_i - \alpha_i - \sum_{n=1}^{N} \phi_{ni})(\Psi'(\gamma_i) - \Psi'(\sum_{j=1}^{k} \gamma_j)) = 0 \\
\Rightarrow & \boxed{\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni}}
\end{aligned}$$

3 Topic Modelling

3.1 LDA model training

In this problem, we perform unsupervised learning on a corpus of news paper articles. We will be performing latent Dirichlet allocation (LDA) using the Mallet tool. The dataset consists of 19,997 articles in 20 known categories (e.g. religion, politics etc.). We implement LDA with 100 topics. The results are explained in the next section.

3.2 Model topics

The model has learned 100 different topics (topic 0 through topic 99). For each of these topics we have the top 20 words that characterize the topics. It is interesting to see how each topic has captured a different text category. We can see how one of the categories is characterized by the words describing its topic. For example, category 13 seems to capture the topic of medicine, with words: ['the', 'and', 'vitamin', 'who', 'candida', 'yeast', 'this', 'for', 'treatment', 'patients', 'has', 'medicine', 'not', 'medical', 'with', 'been', 'weight', 'symptoms', 'doctor', 'can'].

It is interesting to see how, since we are using more topics than article categories, some of the topics seem really similar. For example, it seems like also category 84 is related to medicine, with words: ['and', 'for', 'the', 'medical', 'with', 'disease', 'cancer', 'health', 'patients', 'aids', 'pain', 'this', 'who', 'hiv', 'that', 'drug', 'treatment', 'number', 'were', 'research'].

Furthermore, it is interesting to see that all of the topics have very highly ranked common words that provide little to no information. For example almost all topics have words such as: ['the', 'and', 'this', 'that', 'for', etc.]. This is because these words are really common in the english language, however there is little information to be gained. In order to solve this problem, we can re-train our model, but also including stop-words, such as the words mentioned above in order to ignore them in the model training. Another interesting observation, is that some topics (e.g. topic 6, 30 & 33), are entirely described by general words, such as the stop-words we mentioned above.

Two interesting categories, out of the 20 originally given, are:

1. Category: 'comp.sys.mac.hardware' with topics: [51, 18, 30], where:
   - 51: ['the', 'and', 'with', 'monitor', 'for', 'apple', 'this', 'mhz', 'video', 'ram', 'mac', 'chip', 'bit', 'card', 'simms', 'cpu', 'board', 'has', 'use', 'pin']
   - 18: ['the', 'drive', 'and', 'disk', 'with', 'hard', 'have', 'system', 'problem', 'drives', 'mac', 'floppy', 'that', 'this', 'problems', 'not', 'tape', 'any', 'for', 'disks']
   - 30: ['any', 'for', 'thanks', 'would', 'have', 'anyone', 'know', 'please', 'can', 'there', 'this', 'does', 'help', 'but', 'i'm', 'could', 'like', 'some', 'and', 'looking'] and:
2. Category 'talk.politics.mideast' with topics: [89, 76, 95], where:
   - 89: ['the', 'israel', 'and', 'israeli', 'for', 'that', 'jews', 'not', 'jewish', 'from', 'arab', 'are', 'arabs', 'writes', 'article', 'has', 'there', 'peace', 'were', 'they']
   - 76: ['the', 'and', 'armenian', 'turkish', 'armenians', 'armenia', 'were', 'turks', 'their', 'that', 'for', 'from', 'are', 'with', 'russian', 'people', 'azerbaijan', 'genocide', 'greek', 'there']
   - 95: ['the', 'and', 'muslims', 'that', 'jews', 'turkish', 'genocide', 'for', 'all', 'this', 'war', 'nazi', 'were', 'nazis', 'muslim', 'world', 'people', 'their', 'germany', 'not']

3.3 Parameter variation

We also tried varying the number of topics when training the LDA model. In specific, we tried with number of topics = 75, 50, 25, 15, and 5. We can see that as the number of topics of the model decreases, the more general the topics become. At around 25 topics, we can see that there is a good

mapping between categories and topics, but below that, the topics start becoming over-generalized and full of common words with little meaning and little information. For example, when training the latent Dirichlet allocation model on as little as 5 topics, we got the following five topics:

- Topic 0 with words: ['the', 'and', 'that', 'was', 'they', 'for', 'were', 'have', 'not', 'you', 'with', 'this', 'from', 'are', 'their', 'people', 'had', 'who', 'there', 'all']
- Topic 1 with words: ['the', 'that', 'and', 'you', 'not', 'are', 'this', 'have', 'for', 'but', 'what', 'with', 'your', 'would', 'they', 'one', 'can', 'there', 'all', 'writes']
- Topic 2 with words: ['the', 'and', 'for', 'that', 'you', 'was', 'have', 'but', 'writes', 'with', 'this', 'they', 'article', 'are', 'not', 'out', 'just', 'about', 'would', 'like']
- Topic 3 with words: ['the', 'and', 'for', 'are', 'this', 'that', 'will', 'with', 'from', 'can', 'have', 'which', 'has', 'other', 'space', 'would', 'use', 'not', 'more', 'key']
- Topic 4 with words: ['the', 'and', 'for', 'you', 'with', 'this', 'have', 'that', 'are', 'can', 'from', 'but', 'not', 'any', 'use', 'file', 'will', 'your', 'there', 'get']

4 MovieLens

Experiment Setup

: For this section we will compare multiple methods that solve the matrix estimation problem. We are given a dataset (MovieLens) with various ratings of movies from different users. We split this dataset into two disjoint sets, a training set and a testing set. Then we train our model on the training set and measure its performance by using the testing set. In order to compare our models objectively, we decided to introduce two simple baseline models:
In model A (user average), the prediction of the rating of the pair (user α, movie i) from the testing data, will be the average rating that user α has given from the training data.
In model B (movie average), the prediction of the rating of the pair (user α, movie i) from the testing data, will be the average rating that movie i was has given from the training data.
Additionally, throughout this experiment, we are comparing the models based on two different metrics, accuracy which is the ratio of the ratings predicted correctly after rounding and Root Mean Squared Error (RMSE), which is $\sqrt{\frac{\sum(a-y)^2}{n}}$, where a is the predicted rating and y is the true rating and n is the number of ratings in the testing set. It shows quantitatively how far our predictions are from the ground truth of the testing data.

In general, throughout this experiment for the movies that we have no data (no users have rated this movie), we will just predict the average of all ratings (approx. 3.8).

Our initial results are shown in Table 1.

	Accuracy	RMSE
User Average	0.37	1.034
Movie Average	0.379	0.98

Table 1: Initial accuracy and RMSE results

4.1 Singular Value Thresholding

The first approach we used was Singular Value Thresholding. More specifically, we first scaled the ratings to change their range from [1 .. 5] to [-1 .. 1], we set all unknown ratings of the matrix to 0 after the transformation and then we use the Singular Value Decomposition (SVD) of the matrix as follows:

$$Y = \sum_i \hat{\sigma_i} \hat{u_i} \hat{v_i}^T$$

Then there are two ways to predict new ratings. Both ways involve modifying the vector $\hat{\sigma}$, which is the vector with the singular values of Y.

Hard thresholding sets all values of $\hat{\sigma_i}$ that are less than our threshold τ_h to 0, while soft thresholding sets all values of $\hat{\sigma_i}$ that are less than our threshold τ_s to 0 and reduces the rest of the values by τ_s.

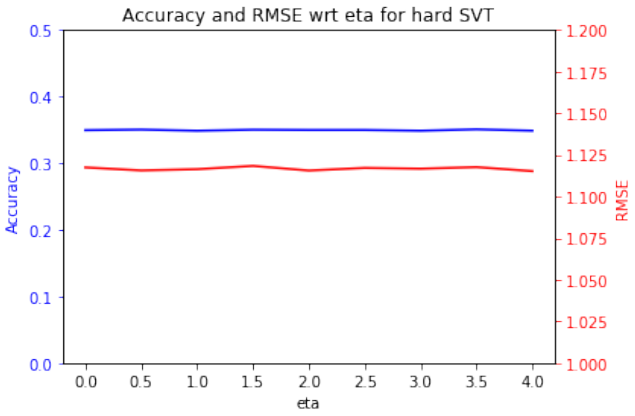

Figure 1: Accuracy and RMSE for Singular Value Thresholding

4.2 Alternative Least Squares

The second approach we used was Alternative Least Squares (ALS). This method is essentially trying to approximate our rating matrix Y by using a product of two low-rank matrices U and V, where $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{k \times n}$. Here we expect k to represent k important features of movies and users that can help our rating prediction.

We then try to minimize the squared loss $||Y - UV||^2$. The way to minimize this is to alternate between fixing U and optimizing with respect to V and vice versa. The optimizing step can actually be viewed a normal linear regression and the closed form formula can be used.

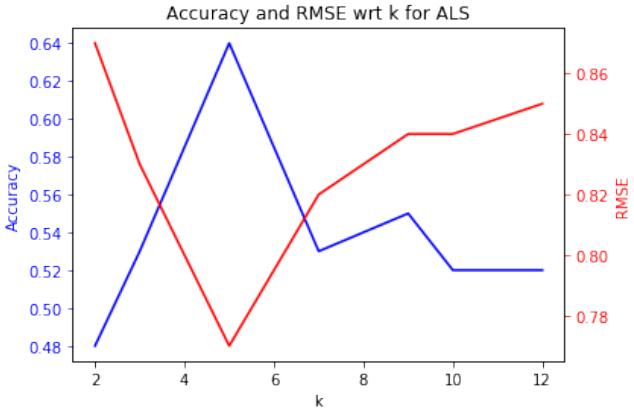

Figure 2: Accuracy and and RMSE vs k

4.3 Collaborative Filtering

One approach that can be used to predict the ratings of the users is to define some similarity metric and aggregate the movie ratings of the k most similar users. We define the similarity of users a and b to be

$$\text{sim}(a,b) = \frac{\langle \hat{Y_a}, \hat{Y_b} \rangle}{\|\hat{Y_a}\| \|\hat{Y_b}\|}$$

where

$$(\hat{Y_a})_j = Y_{a,j} - \frac{1}{M(a,b)} \sum_{l \in M(a,b)} Y_{a,l}.$$

The method we chose for aggregating the chosen users was to take a weighted average of the ratings of the users that have seen the movie. The weights in our case are the similarities of the users with the original user. Since the similarities can be negative, we have to first center the ratings to zero and then center them back to 3.

Then, for k we chose the values 1, 5, 10, 50, 100, 500 and 1000. The results are shown below:

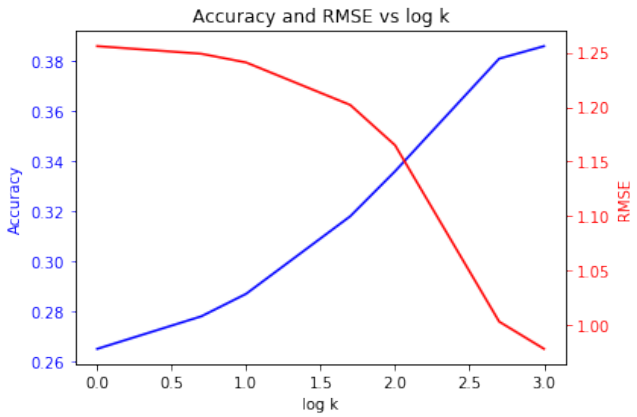

Figure 3: Accuracy and route mean squared error versus log k

We observe that as we choose more similar users, the accuracy increases and the rmse drops.

4.4 Neural Network for

For this section, we use the keras library in order to create a neural network that predicts the rating of a user for a movie. That is, the neural network takes as input the movie ratings, the user features (id, age, sex, postcode) and the movie features (id, genres). We implemented a simple feed forward network.

The network had 3 hidden layers and in the end it was predicting a real value (between 1..5).

We can see that even though neural networks are powerful, they performed worse when it comes to this task. This is because neural networks are very general and tend to overfit as seen in our figure below. For this specific task there are specific solutions to it. For this one we used MSE due to the fact that the software we used (keras) supports it.

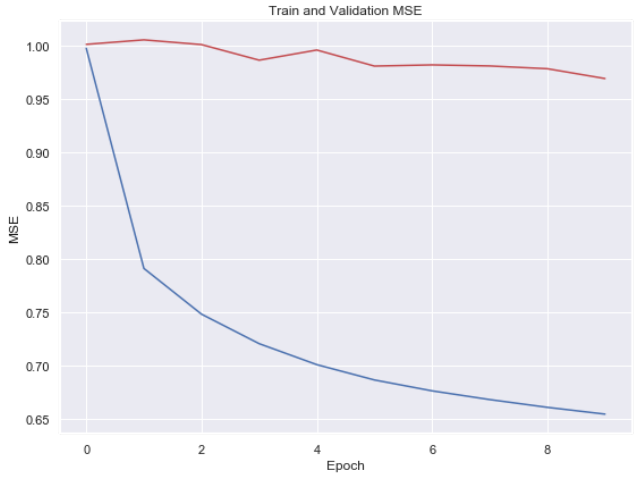

Figure 4: Train(blue) and validation(red) Mean Squared Error

In the end we correctly classified a bit under half the movies in the test set. That means that our accuracy was almost 50%.