EECE 5698 Project:

Movie Recommender System using Collaborative Filtering

Emmanuel Ojuba
Iuliia Klykova

April 18, 2019

Abstract
Recommendation systems are ubiquitous in the current age, with many successful technology companies, such as Amazon and Netflix, relying on these systems to recommend products to their customers. In this project, we explore the collaborative filtering approach to building recommendation systems. We evaluate our method on a subset of The Movies dataset which consists of 26 million actual ratings on a scale of 1 to 5 from users and movies. We also explore the effect of parallelism on the computational speedup of our algorithms. We evaluate our methods using the root mean square error (RMSE) between the actual rating and the predicted rating, with our best performing model yielding an RMSE of 1.125.

Introduction
There are exist different types of recommender systems [1]:

- Demographic Filtering: offers generalized recommendations to every user, based on movie popularity and/or genre. It recommends the same movies to users with similar demographic features.
- Content Based filtering: suggests similar items based on a particular item
- Collaborative Filtering: matches persons with similar interests and provides recommendations based on this matching. It is based on the idea that people who similarly evaluated a certain movie/item in the past are likely to agree again in the future.
- Hybrid Filtering: combines content based and collaborative filtering to build an engine that gave movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user

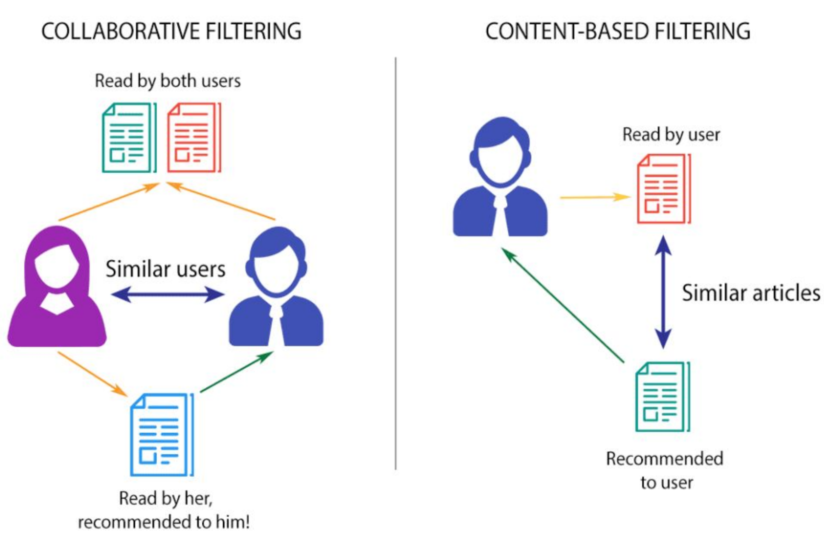

Figure 1. Collaborative Filtering vs Content-Based Filtering

In this project we comprehensively explore collaborative filtering method - matrix factorization.

Matrix Factorization

The matrix factorization approach represents the ratings represented in matrix D, as a product of two matrices, the user matrix U, and the item matrix V. This method assumes the ratings follow a bi-linear relationship, in which the rating given by user i to item j can be represented by the inner product of the user and item profiles u_i and v_j, with some noise [4].

$$r_{ij} = u_i^\top v_j + \varepsilon_{ij}$$

where $u_i, v_j \in \mathbb{R}^d$ are "latent" d-dimensional vectors, and ε_{ij} are i.i.d. random noise variables. In oth
user and item is characterized by a d-dimensional vector (the user and item *profiles*, respectively) such

The goal of this approach is to use the observed ratings (non-sparse elements in D) to infer both U and V. With the inferred U and V, we can make predictions on user-item pairs that did not exist in D as:

We utilize an L2 training objective in matrix factorization given by:

$$\mathsf{RSE}(U, V) = \sum_{(i,j,r_{ij}) \in \mathcal{D}} (u_i^\top v_j - r_{ij})^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \mu \sum_{j=1}^{n} \|v_j\|_2^2,$$

Learning Algorithms

We explore both stochastic gradient descent and alternating least-square methods to optimize this non-convex objective.

<u>Stochastic Gradient Descent</u>

The stochastic gradient descent method randomly initializes the U and V matrices, predicts the ratings for the initialized U and V matrices, and updates these matrices iteratively using a stochastic estimate of the gradient, obtained by calculating the gradient over a subsampled set of the data. The iterations continue until a convergence criteria is met. Stochastic gradient descent is usually performed when the dataset is large, and it will be costly to obtain the true gradient evaluated on the entire dataset:

$$\nabla_{u_1}\widetilde{RSE} = 2 \sum_{v_j, subsampled} \delta_{ij} v_j + 2\lambda u_i$$

$$\nabla_{v_j}\widetilde{RSE} = 2 \sum_{u_i, subsampled} \delta_{ij} u_i + 2\mu v_j$$

where $\boldsymbol{\delta}$ij is the difference between the predicted rating $u_i\, v_j$ and the actual rating .
And then:

$$u^{k+1} = u^k \; - \gamma \nabla_u \widetilde{RSE(U,V)}$$

$$v^{k+1} = v^k \; - \gamma \nabla_v \widetilde{RSE(U,V)}$$

Where γ is a learning rate that can be found for each k-th iteration as:

$$\gamma^k = \frac{\gamma}{k^{power}},$$

Alternating Least Squares

In the alternating least squares method, we rotate between fixing U and V. When U is fixed, the training objective becomes convex with respect to V, and can be solved using evaluating the analytic solution to a least-squares problem [3]. This observation also holds true when V is fixed; U can also be obtained using an analytic solution[5]:

$$v_j = \left[\sum_i u_i u_i^T + \mu I\right]^{-1} \left[\sum_i r_{ij} u_i\right]$$

We utilize Spark MLLib's implementation of alternating least squares, which uses the following parameters:

- **numBlocks**: Number of blocks used to parallelize computation
- **rank**: Latent dimension of u and v
- **iterations:** Number of iterations of ALS to run. ALS typically converges to a reasonable solution in 20 iterations or less.
- **lambda:** The regularization parameter in the optimization
- **nonnegative** – A value of True will solve least-squares with nonnegativity constraints.

We set the nonnegative parameter to True, as all ratings are positive

Dataset [2]

For this project we used The Movie Dataset from Kaggle.com. It contains seven csv file which metadata for all 45,000 movies, data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

For the project we only used **ratings.csv** file that contains 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5. The file is presented in the format: userId, movieId, rating, timestamp. We extracted (userID, movieID, rating) for further processing.

From the dataset we picked 450,000 random samples and randomly divided them into 5 folds that later used for k-fold cross validation.

- Fold 0 with 360000 train samples and 90000 test samples. Training set contains 121650 users and 8611 items.
- Fold 1 with 360000 train samples and 90000 test samples. Training set contains 121573 users and 8613 items
- Fold 2 with 360000 train samples and 90000 test samples. Training set contains 121646 users and 8578 items
- Fold 3 with 360000 train samples and 90000 test samples. Training set contains 121695 users and 8610 items.
- Fold 4 with 360000 train samples and 90000 test samples. Training set contains 121693 users and 8593 items

Results

Stochastic Gradient Descent

In our SGD algorithm, we experimented with various learning rates and settled with a learning rate of 0.0003 with an exponential decay of 0.1, which was provided relatively fast convergence. We used a sample size of 900 to estimate the gradient at each step. For all experimental runs, we made 70 SGD iterations.

Though the values of TestRMSE didn't vary too much for different values of d and λ, the best results TestRMSE = 3.388762 were obtained for d =2 and λ = 10.0.

The Figure 4 shows the dependence of TestRMSE from the regularization parameters for d=2, lambda is set to 0.

d	$\lambda = \mu$	*TestRMSE*
2	10.0	3.388762

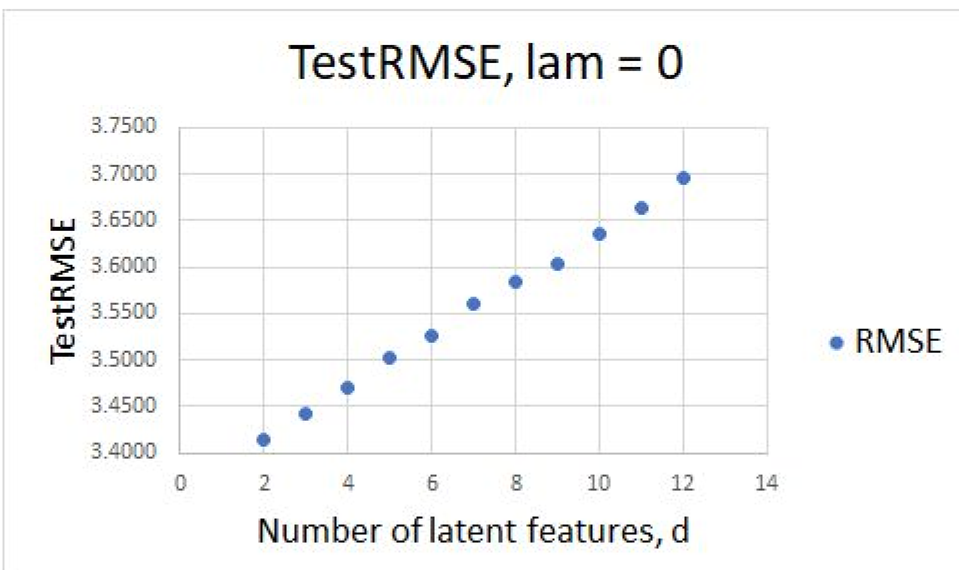

Figure 2. TestRMSE vs. the number of latent features for λ = 0.0

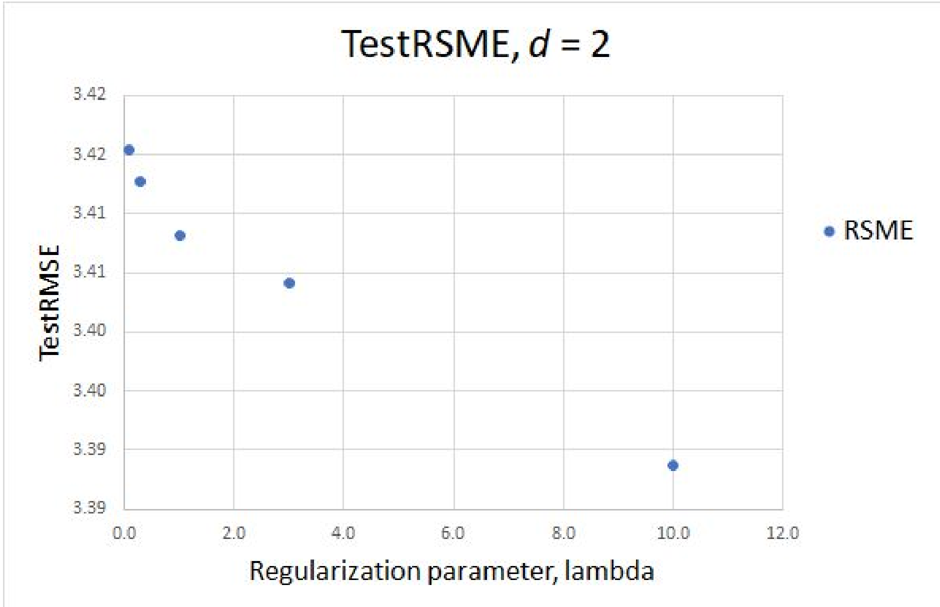

Figure 3. TestRMSE vs. the regularization parameters for $d = 2$
(Detailed Results for all Experimental Runs can be found in the Appendix)

<u>Alternating-Least Squares</u>

In order to find optimal parameters like the number of latent features d and regularization parameters values for both users λ and movies μ we ran a number of experiments.

We changed d in a range from 2 to 12 and $\lambda = \mu$ in the range from 0.1 to 10 and found out that the model performs best for the small values of d. The smallest TestRMSE were observed for:

d	$\lambda = \mu$	*TestRMSE*
3	0.3	1.1248

Figure 4 shows the dependence of TestRMSE from number of latent features for $\lambda \approx 0$.

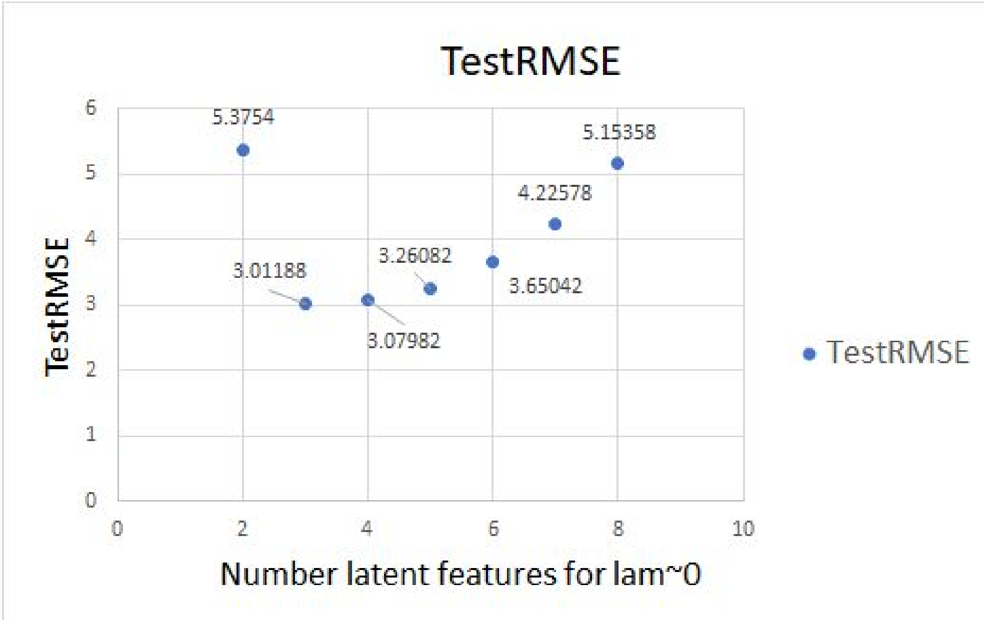

Figure 4. TestRMSE vs. number of latent features for λ = μ ≃ 0

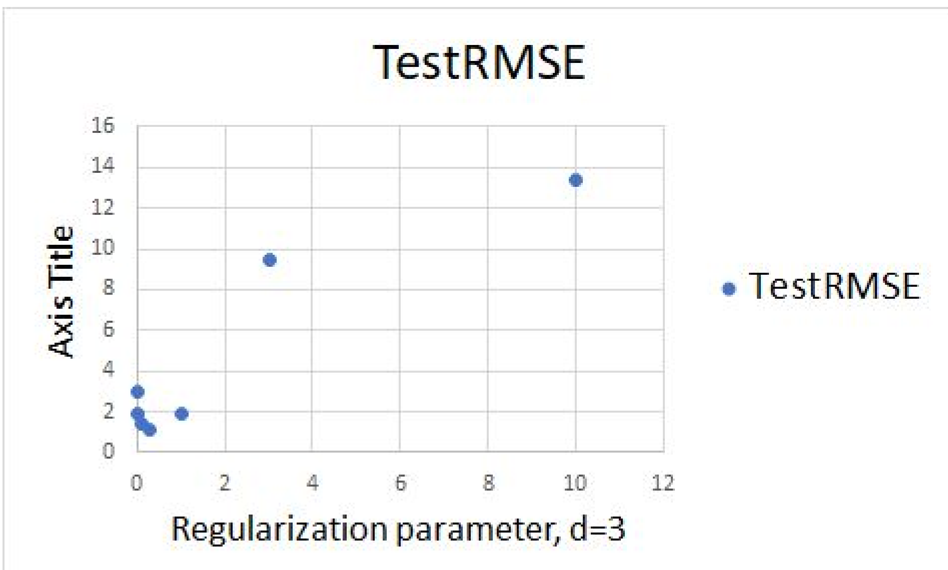

Figure 5. TestRMSE vs.Regularization parameter for *d*= 3
(Detailed Results for all Experimental Runs can be found in the Appendix)

Figure 5 shows the dependence of TestRMSE from the regularization parameters for *d*=3.

Overall, the best performing model was the ALS-optimized matrix factorization with a latent dimension of 3 and regularization parameter of 0.01, which yielded a cross-validation RMSE of 1.125.

Parallelism and Performance

For our best performing model, the alternating least squares-optimized matrix factorization with a latent dimension of 3, we explore the effect of parallelism by altering the numBlocks parameter, which controls the number of computing blocks used to parallelize the operation. From our results displayed below, it is clear that increasing the parallelism of the algorithm produces a computational speedup.

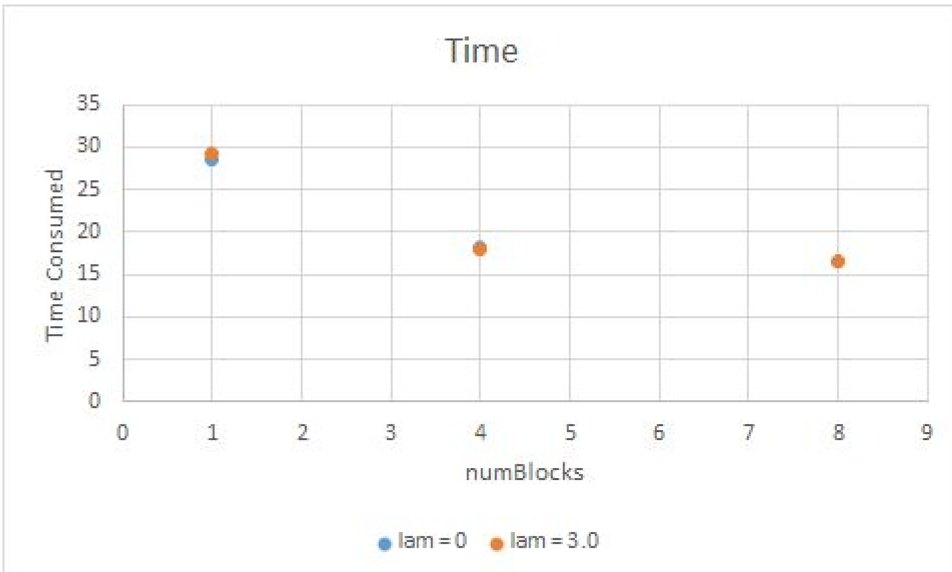

Figure 6: Effect of Parallelism on the Speed of ALS Matrix Factorization

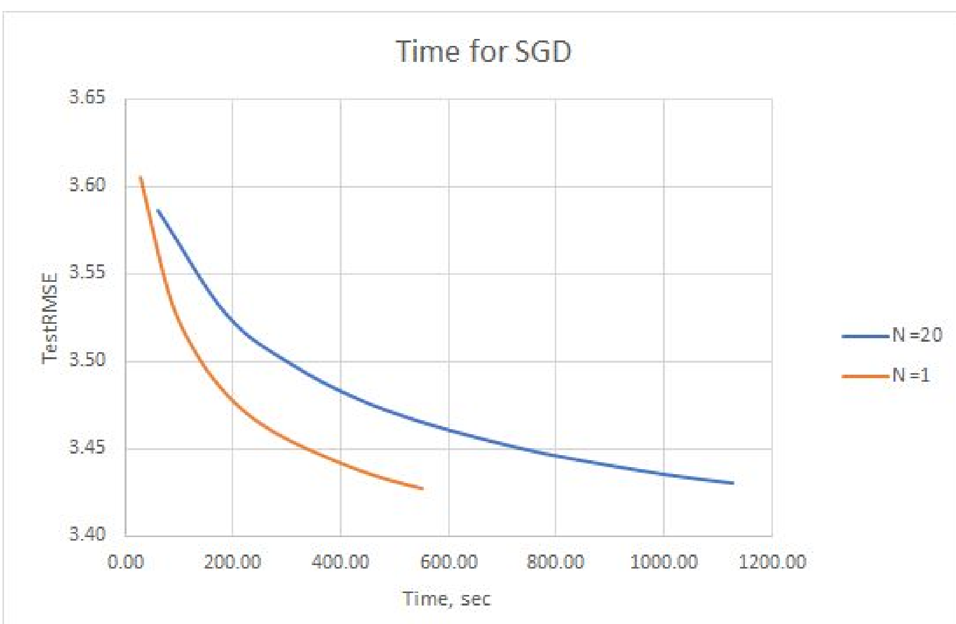

Figure 7: Effect of Parallelism on the Speed of SGD Matrix Factorization

We also explored the effect of parallelism on the SGD-optimized algorithm. We did not observe any speedup in the computational time, likely due to code inefficiencies. One of such bottlenecks could the that training process did involve computing the RMSE over the entire training and test sets at each iteration.

Conclusion

In this project we implemented a movie recommender system using Collaborative filtering approach. We explored the stochastic gradient descent and alternating least-square as optimization methods and we were able to achieve an RMSE of 1.125

References

[1] - Ahmed I. Getting Started with a Movie Recommendation System. https://www.kaggle.com/ibtesama/getting-started-with-a-movie-recommendation-system/notebook

[2] - The Movies Dataset https://www.kaggle.com/rounakbanik/the-movies-dataset

[3] - Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems

[4] - Ioannidis, E. Homework 4 Handout

[5] - Matrix Completion via Alternating Least Squares - http://stanford.edu/~rezab/classes/cme323/S15/notes/lec14.pdf

[6] - Various Implementations of Collaborative Filtering https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

Work Split

Both team members, Iuliia and Emmanuel contributed fairly to the project. Emmanuel performed the duties of ALS coding, SGD coding, parallelism comparison, research and report preparation. Iuliia performed the duties of SGD coding, parallelism comparison, plot creation, research and report preparation.

Appendix A. Outputs for ALS

Train RMSE	Test RMSE	lam	numIterations	rank	k0	k1	k2	k3	k4	Average
4.459	29.0787	0.01	7	2	29.0787	28.9834	31.1462	26.9762	29.5792	29.15274
2.3225	24.6756	0.01	7	3	24.6756	36.5149	18.1296	25.8201	20.2573	25.0795
0.8855	15.0104	0.01	7	4	15.0104	21.7792	18.4855	30.0955	14.6265	19.99942
0.9826	20.5877	0.01	7	5	20.5877	19.0373	24.8899	32.2887	35.451	26.45092
1.0791	28.6478	0.01	7	6	28.6478	25.6072	27.5231	30.2827	25.8112	27.5744
0.5379	20.1394	0.01	7	7	20.1394	18.0732	22.1212	35.2634	20.8688	23.2932
0.5715	26.9215	0.01	7	8	26.9215	27.2308	25.4757	26.9457	32.0935	27.73344
1.5712	17.7301	0.01	10	2	17.7301	18.8515	21.8219	16.2254	17.3296	18.3917
0.5768	8.6102	0.01	10	3	8.6102	25.647	6.808	9.92	7.5716	11.71136
0.3068	5.7824	0.01	10	4	5.7824	7.8887	6.5836	13.4908	6.1331	7.97572
0.3104	7.8108	0.01	10	5	7.8108	6.806	9.8324	14.8637	17.9095	11.44448
0.4292	13.1783	0.01	10	6	13.1783	11.209	11.8318	14.5159	10.8389	12.31478
0.2094	9.6712	0.01	10	7	9.6712	8.7704	10.7166	23.6489	10.3666	12.63474
0.2816	15.6263	0.01	10	8	15.6263	16.0468	15.4422	15.254	21.4611	16.76608
0.5922	9.3575	0.01	13	2	9.3575	10.958	11.1733	10.6379	8.8783	10.201
0.3026	3.9599	0.01	13	3	3.9599	10.3761	4.0325	4.7697	4.2757	5.48278
0.2008	3.633	0.01	13	4	3.633	4.4703	3.7748	5.6767	4.332	4.37736
0.1708	4.5203	0.01	13	5	4.5203	4.236	5.0761	6.0753	6.9486	5.37126

0.1809	6.5046	0.01	13	6	6.5046	6.0316	5.8643	6.9444	5.7178	6.21254
0.1128	6.1144	0.01	13	7	6.1144	5.7094	6.4962	13.2096	6.3817	7.58226
0.1366	9.3999	0.01	13	8	9.3999	9.5688	9.4893	8.986	13.2251	10.13382
0.3862	5.2344	0.01	20	2	5.2344	5.6119	5.2456	5.8673	4.9178	5.3754
0.2262	2.8147	0.01	20	3	2.8147	3.4965	2.9016	2.9268	2.9198	3.01188
0.1523	2.8897	0.01	20	4	2.8897	3.0018	2.7677	3.331	3.4089	3.07982
0.1136	3.232	0.01	20	5	3.232	3.2097	3.2643	3.2332	3.3649	3.26082
0.0901	3.6952	0.01	20	6	3.6952	3.7548	3.4921	3.6972	3.6128	3.65042
0.0626	4.0193	0.01	20	7	4.0193	3.9243	4.0564	5.1059	4.023	4.22578
0.0542	4.9698	0.01	20	8	4.9698	4.9728	5.0448	4.8399	5.9406	5.15358
2.2001	11.3669	0.03	7	2	11.3669	11.2257	14.6952	8.6376	11.6351	11.5121
0.6023	4.6294	0.03	7	3	4.6294	15.3494	3.3482	5.1603	3.4976	6.39698
0.3143	3.199	0.03	7	4	3.199	4.2231	3.6174	6.8149	3.1827	4.20742
0.2942	4.2087	0.03	7	5	4.2087	4.0233	5.0874	7.7015	8.7422	5.95262
0.3439	6.5127	0.03	7	6	6.5127	5.7574	5.7866	6.9026	5.5586	6.10358
0.1924	5.0662	0.03	7	7	5.0662	4.811	5.4424	11.2735	5.3053	6.37968
0.2151	7.1126	0.03	7	8	7.1126	7.3877	7.1725	6.9491	9.4908	7.62254
0.4511	2.5167	0.03	10	2	2.5167	2.761	3.0898	2.4467	2.5743	2.6777
0.2772	2.0295	0.03	10	3	2.0295	3.1379	1.9981	2.0179	1.9678	2.23024
0.1998	2.257	0.03	10	4	2.257	2.3157	2.2361	2.502	2.2117	2.3045
0.1625	2.612	0.03	10	5	2.612	2.5895	2.6788	2.953	3.0309	2.77284
0.1433	3.1386	0.03	10	6	3.1386	3.0935	2.9641	3.2095	3.0525	3.09164
0.1038	3.219	0.03	10	7	3.219	3.1458	3.2742	4.3745	3.2245	3.4476
0.0947	3.8962	0.03	10	8	3.8962	3.9417	3.9298	3.8004	4.4562	4.00486
0.3481	1.8536	0.03	13	2	1.8536	1.8596	1.8235	2.0195	1.8799	1.88722

0.2386	1.8935	0.03	13	3	1.8935	1.9968	1.8977	1.8764	1.9032	1.91352
0.1721	2.1729	0.03	13	4	2.1729	2.1569	2.1243	2.1769	2.1456	2.15532
0.1328	2.3938	0.03	13	5	2.3938	2.386	2.371	2.4207	2.4301	2.40032
0.1068	2.6096	0.03	13	6	2.6096	2.6391	2.532	2.6223	2.6212	2.60484
0.0796	2.756	0.03	13	7	2.756	2.7286	2.7542	3.0317	2.7367	2.80144
0.0668	3.0642	0.03	13	8	3.0642	3.0669	3.0376	3.0053	3.2222	3.07924
0.3167	1.8435	0.03	20	2	1.8435	1.8004	1.7535	1.9275	1.7915	1.82328
0.2122	1.9255	0.03	20	3	1.9255	1.9371	1.8999	1.9194	1.9309	1.92256
0.1485	2.153	0.03	20	4	2.153	2.1146	2.1091	2.1205	2.1416	2.12776
0.1078	2.2892	0.03	20	5	2.2892	2.2724	2.2579	2.2665	2.2625	2.2697
0.0809	2.3602	0.03	20	6	2.3602	2.3849	2.3299	2.3473	2.3727	2.359
0.0597	2.4309	0.03	20	7	2.4309	2.4192	2.4028	2.4391	2.4153	2.42146
0.0462	2.4942	0.03	20	8	2.4942	2.4885	2.4552	2.4836	2.5116	2.48662
0.6039	2.5445	0.1	7	2	2.5445	2.8105	3.3951	2.0301	2.5078	2.6576
0.3328	1.6288	0.1	7	3	1.6288	2.6687	1.5547	1.6037	1.5728	1.80574
0.263	1.6723	0.1	7	4	1.6723	1.6799	1.6749	1.8253	1.6543	1.70134
0.2258	1.8042	0.1	7	5	1.8042	1.7846	1.8363	1.9818	2.0055	1.88248
0.2003	1.9702	0.1	7	6	1.9702	1.9298	1.8717	1.9592	1.9082	1.92782
0.1725	1.9411	0.1	7	7	1.9411	1.9172	1.932	2.1611	1.9296	1.9762
0.1579	2.0661	0.1	7	8	2.0661	2.0602	2.019	2.043	2.1361	2.06488
0.3709	1.4033	0.1	10	2	1.4033	1.383	1.3693	1.3644	1.3726	1.37852
0.2794	1.4809	0.1	10	3	1.4809	1.5008	1.4606	1.4663	1.4713	1.47598
0.2228	1.5673	0.1	10	4	1.5673	1.5491	1.5517	1.5723	1.5628	1.56064
0.187	1.6389	0.1	10	5	1.6389	1.616	1.6269	1.6413	1.6502	1.63466
0.1623	1.6742	0.1	10	6	1.6742	1.6722	1.6404	1.665	1.6665	1.66366

0.1414	1.7037	0.1	10	7	1.7037	1.6874	1.6788	1.7048	1.6848	1.6919
0.1278	1.7204	0.1	10	8	1.7204	1.711	1.6892	1.7184	1.7294	1.71368
0.3484	1.3742	0.1	13	2	1.3742	1.3265	1.3176	1.3428	1.3367	1.33956
0.2605	1.4531	0.1	13	3	1.4531	1.4421	1.4314	1.4425	1.4446	1.44274
0.2055	1.5178	0.1	13	4	1.5178	1.5004	1.5041	1.5159	1.5184	1.51132
0.1701	1.5743	0.1	13	5	1.5743	1.5482	1.558	1.5581	1.5668	1.56108
0.1458	1.5854	0.1	13	6	1.5854	1.5842	1.5643	1.5759	1.5823	1.57842
0.1271	1.6125	0.1	13	7	1.6125	1.5962	1.5851	1.5862	1.589	1.5938
0.1141	1.5998	0.1	13	8	1.5998	1.5938	1.5778	1.6067	1.6008	1.59578
0.3326	1.3624	0.1	20	2	1.3624	1.3122	1.2977	1.3188	1.3182	1.32186
0.244	1.4114	0.1	20	3	1.4114	1.4048	1.3872	1.4048	1.4072	1.40308
0.1888	1.4526	0.1	20	4	1.4526	1.4428	1.4418	1.4505	1.4595	1.44944
0.1532	1.5024	0.1	20	5	1.5024	1.4712	1.4791	1.4778	1.4869	1.48348
0.1293	1.4947	0.1	20	6	1.4947	1.4996	1.4855	1.4907	1.4978	1.49366
0.1123	1.5195	0.1	20	7	1.5195	1.4974	1.493	1.4857	1.4899	1.4971
0.0999	1.4883	0.1	20	8	1.4883	1.4866	1.4732	1.5034	1.4878	1.48786
0.5265	1.4792	0.3	7	2	1.4792	1.7399	1.6434	1.3656	1.4548	1.53658
0.4693	1.2585	0.3	7	3	1.2585	1.4726	1.2174	1.2317	1.2278	1.2816
0.4444	1.2189	0.3	7	4	1.2189	1.2153	1.2166	1.2633	1.2169	1.2262
0.4275	1.2435	0.3	7	5	1.2435	1.224	1.2392	1.28	1.2854	1.25442
0.4137	1.2607	0.3	7	6	1.2607	1.2426	1.2372	1.2498	1.2499	1.24804
0.4082	1.2486	0.3	7	7	1.2486	1.2306	1.2357	1.2685	1.2419	1.24506
0.3976	1.2542	0.3	7	8	1.2542	1.2422	1.237	1.2546	1.2672	1.25104
0.4955	1.1706	0.3	10	2	1.1706	1.162	1.1532	1.1486	1.165	1.15988
0.4536	1.1797	0.3	10	3	1.1797	1.1735	1.161	1.1683	1.1695	1.1704

0.4273	1.1828	0.3	10	4	1.1828	1.1725	1.1718	1.1784	1.1805	1.1772
0.4126	1.1984	0.3	10	5	1.1984	1.1798	1.1834	1.1907	1.1981	1.19008
0.4006	1.1989	0.3	10	6	1.1989	1.1912	1.1845	1.1931	1.196	1.19274
0.3929	1.2047	0.3	10	7	1.2047	1.193	1.1896	1.1948	1.1989	1.1962
0.3833	1.2015	0.3	10	8	1.2015	1.1926	1.1888	1.2017	1.2029	1.1975
0.486	1.1574	0.3	13	2	1.1574	1.1353	1.1369	1.1408	1.153	1.14468
0.4434	1.174	0.3	13	3	1.174	1.1606	1.1557	1.1636	1.1648	1.16374
0.4167	1.1774	0.3	13	4	1.1774	1.1662	1.1648	1.1679	1.1755	1.17036
0.4029	1.1919	0.3	13	5	1.1919	1.1724	1.176	1.1803	1.1887	1.18186
0.3902	1.1898	0.3	13	6	1.1898	1.1838	1.1769	1.1846	1.1877	1.18456
0.3827	1.1963	0.3	13	7	1.1963	1.1867	1.1818	1.1835	1.1907	1.1878
0.373	1.1912	0.3	13	8	1.1912	1.1832	1.1793	1.1921	1.1917	1.1875
0.4784	1.1588	0.3	20	2	1.1588	1.1335	1.1371	1.142	1.1513	1.14454
0.4334	1.1728	0.3	20	3	1.1728	1.1602	1.1539	1.1636	1.1671	1.16352
0.4054	1.1733	0.3	20	4	1.1733	1.1622	1.1602	1.1619	1.1732	1.16616
0.3922	1.1877	0.3	20	5	1.1877	1.1668	1.1731	1.1755	1.1851	1.17764
0.379	1.1845	0.3	20	6	1.1845	1.1792	1.1722	1.1785	1.1844	1.17976
0.3712	1.1901	0.3	20	7	1.1901	1.1823	1.177	1.1766	1.186	1.1824
0.3633	1.1863	0.3	20	8	1.1863	1.177	1.1735	1.1856	1.1863	1.18174
1.4802	2.0088	1	7	2	2.0088	2.1513	2.043	1.9659	1.9889	2.03158
1.4772	1.9514	1	7	3	1.9514	2.0607	1.9315	1.9362	1.9399	1.96394
1.4776	1.9278	1	7	4	1.9278	1.9143	1.9244	1.9472	1.9251	1.92776
1.4772	1.9338	1	7	5	1.9338	1.9108	1.9316	1.9569	1.964	1.93942
1.4765	1.9423	1	7	6	1.9423	1.9198	1.9263	1.9358	1.9409	1.93302
1.4772	1.9301	1	7	7	1.9301	1.9111	1.9249	1.9475	1.9322	1.92916

1.4766	1.9364	1	7	8	1.9364	1.9187	1.9255	1.9302	1.9483	1.93182
1.479	1.9165	1	10	2	1.9165	1.9029	1.9075	1.9074	1.9152	1.9099
1.4792	1.9141	1	10	3	1.9141	1.9011	1.9031	1.9057	1.9124	1.90728
1.4792	1.9113	1	10	4	1.9113	1.8948	1.9025	1.9055	1.9098	1.90478
1.4792	1.9118	1	10	5	1.9118	1.8917	1.9032	1.9056	1.9156	1.90558
1.4792	1.9122	1	10	6	1.9122	1.8942	1.9013	1.9046	1.9139	1.90524
1.4792	1.91	1	10	7	1.91	1.8938	1.9014	1.905	1.9129	1.90462
1.4792	1.911	1	10	8	1.911	1.8954	1.9025	1.9054	1.9124	1.90534
1.4794	1.9126	1	13	2	1.9126	1.8947	1.9026	1.9048	1.9108	1.9051
1.4794	1.9119	1	13	3	1.9119	1.8959	1.9015	1.9041	1.911	1.90488
1.4794	1.9101	1	13	4	1.9101	1.8935	1.9012	1.9036	1.9087	1.90342
1.4794	1.9102	1	13	5	1.9102	1.8904	1.9013	1.903	1.913	1.90358
1.4794	1.9104	1	13	6	1.9104	1.8927	1.8998	1.9028	1.9122	1.90358
1.4794	1.9086	1	13	7	1.9086	1.8926	1.8999	1.9028	1.9115	1.90308
1.4794	1.9094	1	13	8	1.9094	1.8938	1.901	1.9039	1.9102	1.90366
1.4795	1.912	1	20	2	1.912	1.894	1.9019	1.9045	1.9096	1.9044
1.4795	1.9114	1	20	3	1.9114	1.8952	1.9012	1.9039	1.9107	1.90448
1.4795	1.9098	1	20	4	1.9098	1.8932	1.901	1.9033	1.9085	1.90316
1.4795	1.9098	1	20	5	1.9098	1.8901	1.9009	1.9026	1.9126	1.9032
1.4795	1.91	1	20	6	1.91	1.8923	1.8996	1.9025	1.9118	1.90324
1.4795	1.9082	1	20	7	1.9082	1.8922	1.8996	1.9025	1.9112	1.90274
1.4795	1.909	1	20	8	1.909	1.8935	1.9007	1.9037	1.9099	1.90336
13.5646	13.4266	3	7	2	13.4266	13.3846	13.4319	13.3971	13.4604	13.42012
13.5488	13.4129	3	7	3	13.4129	13.3889	13.4041	13.3911	13.4291	13.40522

13.5166	13.3848	3	7	4	13.3848	13.3479	13.4036	13.4019	13.411	13.38984
13.5325	13.3982	3	7	5	13.3982	13.3315	13.4142	13.4051	13.4631	13.40242
13.539	13.4045	3	7	6	13.4045	13.3652	13.4006	13.3988	13.4441	13.40264
13.524	13.3913	3	7	7	13.3913	13.3388	13.4028	13.4036	13.4218	13.39166
13.5447	13.4098	3	7	8	13.4098	13.364	13.4024	13.3784	13.4563	13.40218
13.4853	13.3558	3	10	2	13.3558	13.3244	13.3965	13.2191	13.3374	13.32664
13.3038	13.1944	3	10	3	13.1944	13.3751	13.08	13.1505	12.999	13.1598
12.9823	12.9045	3	10	4	12.9045	12.9167	13.0703	13.273	12.8281	12.99852
13.1334	13.0396	3	10	5	13.0396	12.763	13.1893	13.3139	13.3681	13.13478
13.2034	13.1041	3	10	6	13.1041	13.0996	13.0457	13.2412	13.1581	13.12974
13.0523	12.968	3	10	7	12.968	12.8312	13.0691	13.2943	12.9316	13.01884
13.2655	13.1607	3	10	8	13.1607	13.0823	13.0627	13.023	13.2922	13.12418
12.7037	12.6484	3	13	2	12.6484	12.7007	12.9977	11.8715	12.2691	12.49748
11.6605	11.676	3	13	3	11.676	13.2049	11.2515	11.5582	10.9828	11.73468
10.768	10.8263	3	13	4	10.8263	10.8945	11.2218	12.1714	10.6624	11.15528
11.0975	11.1415	3	13	5	11.1415	10.6046	11.6464	12.4524	12.4792	11.66482
11.3003	11.3348	3	13	6	11.3348	11.4218	11.1521	11.9881	11.4273	11.46482
10.9085	10.9609	3	13	7	10.9609	10.7224	11.219	12.312	10.842	11.21126
11.5131	11.5371	3	13	8	11.5371	11.3592	11.2	11.1346	12.0074	11.44766

9.3851	9.489	3	20	2	9.489	9.4546	9.5308	9.4479	9.4983	9.48412
9.3549	9.4585	3	20	3	9.4585	9.6137	9.4537	9.4432	9.4795	9.48972
9.3471	9.4506	3	20	4	9.4506	9.4104	9.4532	9.4543	9.4774	9.44918
9.3493	9.4528	3	20	5	9.4528	9.408	9.4576	9.464	9.5058	9.45764
9.351	9.4545	3	20	6	9.4545	9.4144	9.4524	9.4497	9.484	9.451
9.348	9.4509	3	20	7	9.4509	9.4094	9.4529	9.4583	9.4792	9.45014
9.3531	9.4562	3	20	8	9.4562	9.4145	9.453	9.4393	9.4912	9.45084
13.5768	13.4331	10	7	2	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	3	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	4	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	5	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	6	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	7	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	7	8	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	2	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	3	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	4	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	5	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	6	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	10	7	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.433	10	10	8	13.433	13.3901	13.435	13.4132	13.4715	13.42858

	1				1					
13.5768	13.4331	10	13	2	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	3	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	4	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	5	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	6	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	7	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	13	8	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	2	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	3	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	4	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	5	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	6	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	7	13.4331	13.3901	13.435	13.4132	13.4715	13.42858
13.5768	13.4331	10	20	8	13.4331	13.3901	13.435	13.4132	13.4715	13.42858

Appendix B. Output of SGD

lam	d	RMSE		lam	d	RMSE
1	7	3.53717		0.0	2	3.4150
10	3	3.402246		0.0	3	3.4430

0.3	10	3.621641		0.0	4	3.4706
0.1	9	3.586382		0.0	5	3.5016
0.1	5	3.492622		0.0	6	3.5264
3	3	3.42708		0.0	7	3.5609
0.1	3	3.442047		0.0	8	3.5838
3	9	3.553926		0.0	9	3.6040
10	4	3.415561		0.0	10	3.6362
1	9	3.571489		0.0	11	3.6627
0.3	7	3.552795		0.0	12	3.6967
3	7	3.516373				
3	8	3.530935				
1	8	3.559074				
1	10	3.610383				
10	7	3.460396				
0.3	12	3.675349				
10	5	3.428554				
3	10	3.578956				
1	4	3.458389				
1	3	3.434021				
10	8	3.470594				
3	4	3.443321				
1	2	3.408139				
0.3	4	3.465081				
0.1	8	3.572637				
0.3	9	3.58546				

0.1	11	3.641897				
3	5	3.468965				
10	2	3.388762				
0.3	5	3.490206				
0.3	2	3.41278				
0.1	6	3.517654				
3	6	3.490007				
0.1	7	3.549543				
0.1	10	3.62988				
0.1	4	3.46273				
0.3	6	3.513954				
0.3	3	3.438089				
0.3	11	3.632808				
0.3	8	3.564828				
10	6	3.444012				
3	2	3.404181				
0.1	2	3.415382				
1	6	3.507796				