Collaborative Filtering via Ensemble of Matrix Factorization & Autoencoder Models

Andreas Opedal, Federico van Swaaij, Mian Zhong, Matúš Žilinec
Group: Unstable Geniuses, Department of Computer Science, ETH Zürich, Switzerland

***Abstract*—Collaborative filtering aims to capture patterns of a large sparse matrix, one major application area being user-item rating predictions for recommender systems. In this paper, we explore extensions involving heuristics-based parameter initialization and SVD imputing to the classical matrix factorization techniques SVD++ and pLSA respectively. We also adapt the more recent variational autoencoder model to a regression setting, before combining all three models in an ensemble. The proposed model compares favorably to a number of baseline models and experiments give insights into the importance of properly tuning hyperparameters like annealing and noise.**

I. Introduction

Having a sophisticated recommender system is a key driver in the commercial success of any online service or product provider. Whether it be an entertainment platform such as Spotify, a social network such as Facebook or an e-commerce platform such as Amazon, they all use item recommendations of various kinds to keep users engaged with their content and prevent customer churn.

A widely used approach to recommender systems is collaborative filtering. The collaborative filtering setting is commonly framed as a matrix consisting of item ratings made by users. Each user has only rated a subset of the available items however. The problem is then to estimate the missing values of the rating matrix, which most often far outnumber those of the observed. Essential in this task is the underlying structural assumption that the reconstructed matrix should be of low rank, which means e.g. that users who have liked similar items in the past are likely to do so also in the future. The classical approach to collaborative filtering has been that of algorithms involving the singular value decomposition [1], [2], which increased in popularity after their success in the Netflix competition of 2006 [3]. In recent years however, research has shifted more towards techniques based on neural networks [4], [5].

In this paper we combine both classical and modern approaches in the form of an ensemble model, which is known to decrease the error of the predictor when the base models are sufficiently diverse [6]. We expand the well-known SVD++ algorithm with learning rate decay, momentum and heuristic-based parameter initialization. As for the more modern techniques, we apply a variational autoencoder on the collaborative filtering setting, which turns out to give better performance with the absence of the KL-divergence term usually found in the loss function. Furthermore, we include a pLSA model extended with SVD for increased stability before we average the predictions of the three models in an ensemble. The ensemble is compared to a number of baseline models and performs favorably in comparison to most of them.

The paper is structured as follows: In Section II the models used in the ensemble and the baseline models are introduced. In Section III we describe the setup and our approach to generate predictions. In Section IV the results of the experiments are presented, which are then discussed in Section V. Finally, we summarize the paper in Section VI.

II. Models

The results of the ensemble model are compared to the performance of its contained base models, as well as a number of additional baseline models. These are all presented in this section. We denote the true, partly observed matrix by $\mathbf{X} \in \mathbb{R}^{m \times n}$ and the reconstructed matrix by $\hat{\mathbf{X}} \in \mathbb{R}^{m \times n}$, where m is the number of users and n is the number of items. Furthermore, let Ω be the set of observed entries.

A. Singular Value Decomposition (SVD)

SVD is a matrix factorization technique that decomposes any arbitrary matrix in the following form

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and $\mathbf{D}$ is a diagonal matrix consisting of the singular values. We start by imputing the missing values with the item-wise sample means. We then perform an SVD of the user/item-matrix, followed by a reconstruction using the k (with $0 < k \leq \min(m, n)$) largest singular values resting on the underlying assumption that $\mathbf{X}$ is of low rank.

B. Alternating Least Squares (ALS)

The goal of ALS is to minimize the reconstruction error subject to a rank constraint on $\mathbf{X}$, using the decomposition:

$$\mathbf{X} = \mathbf{P}^T\mathbf{Q} \tag{2}$$

where $\mathbf{P} \in \mathbb{R}^{k \times m}, \mathbf{Q} \in \mathbb{R}^{k \times n}$ for some integer k with $0 < k \leq \min(m, n)$. In our setting, given a user u and an item i, we assume that each row vector $\mathbf{p}_u^T$ in $\mathbf{P}^T$ and each column vector $\mathbf{q}_i$ in $\mathbf{Q}$ represents the user and the item respectively in a low-dimensional vector space. The estimate of a rating

is given by the inner product $\hat{x}_{ui} = \mathbf{p}_u^T \mathbf{q}_i$ that minimizes the following non-convex least squares objective:

$$\sum_{(u,i)\in\Omega} (x_{ui} - \hat{x}_{ui})^2 + \lambda(\sum_u |N(u)| \cdot ||\mathbf{p}_u||^2 + \sum_i |N(i)| \cdot ||\mathbf{q}_i||^2) \tag{3}$$

where $N(u)$ and $N(i)$ denote the set of all ratings of user u and item i respectively[1], while λ tunes the strength of the regularization. The ALS algorithm uses the convexity in $\mathbf{P}^T$ and $\mathbf{Q}$ individually by minimizing the loss with respect to $\mathbf{P}$ while keeping $\mathbf{Q}$ fixed, and vice versa. This procedure is repeated out until convergence.

C. Singular Value Thresholding (SVT)

As opposed to ALS, the SVT algorithm [2] aims to approximate the exact matrix reconstruction problem, where the rank of the recovered matrix is minimized subject to exact recovery of the observed entries. This is accomplished by a convex relaxation of the rank objective with the nuclear norm, resulting in the following optimization problem:

$$\begin{aligned} \min_{\mathbf{Z}\in\mathbb{R}^{m\times n}} \quad & ||\mathbf{Z}||_* \\ \text{subject to} \quad & Z_{ui} = X_{ui} \quad \forall (u,i) \in \Omega \end{aligned} \tag{4}$$

The algorithm applies the following operations iteratively until a stopping criterion is reached:

$$\begin{aligned} \mathbf{Z}^k &= \mathcal{D}_\tau(\mathbf{Y}^{k-1}, \tau) \\ \mathbf{Y}^k &= \mathbf{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\mathbf{A} - \mathbf{Z}^k) \end{aligned} \tag{5}$$

with $\mathcal{D}_\tau(\mathbf{Y}, \tau) := \mathbf{U}(\text{diag}\{(\sigma_j - \tau)_+\})\mathbf{V}^T$, $\mathcal{P}_\Omega$ being an operator keeping only the values in Ω as nonzero, and δ_k being a sequence of stepsizes. This algorithm converges to the solution of a problem closely related to (4) for choices of δ_k such that $0 < \inf \delta_k \leq \sup \delta_k < 2$, which in turn approximates (4) well for large values of τ.

D. Probabilistic Latent Semantic Analysis (pLSA)

PLSA [8] is a co-occurence model that introduces latent factors with hidden states for every co-occurrence pair, to decompose the conditional probability of the variable of interest. The Gaussian distribution is used to describe the variable of interest conditional on latent factors. The probability of rating x from the user-item pair (u, i) via latent variables z is calculated as,

$$p(x|u,i) = \sum_z P(z|u) p(x; \mu_{i,z}, \sigma_{i,z}) \tag{6}$$

where $x_{i,z} \sim \mathcal{N}(\mu_{i,z}, \sigma_{i,z})$ is assumed. The estimated rating is generated by taking the expectation in the following way,

$$\hat{x}_{ui} = \mathbb{E}[x|u,i] = \int_{\mathcal{X}} x p(x|u,i) dx = \sum_z P(z|u) \mu_{i,z} \tag{7}$$

where we apply the EM-algorithm to find estimates for $P(z|u)$ and $\mu_{i,z}$.

[1]The addition of $N(u)$ and $N(i)$ is an extension of the regular model from [7].

E. SVD++

Similar to the setup of ALS, SVD++ [1] also considers a low-rank representation of the observed data. In addition, it introduces bias terms for the items and users as well as information on the user implicit feedback. The predictions of a rating are outputted as follows:

$$\hat{x}_{ui} = \mu + b_i + b_u + \mathbf{q}_i^T \left(\mathbf{p}_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} \mathbf{y}_j \right) \tag{8}$$

where b_i and b_u represent the bias of item i and user u respectively, $\sum \mathbf{y}_j$ represents the sum of item factors for user u and $N(u)$ denotes the set of items rated by user u. The loss function takes the mean squared error over the ratings and adds five l_2-regularization penalties ($\lambda_1, \lambda_2, \lambda_3, \lambda_P$ and λ_Q) for the parameter vectors, and is commonly minimized with stochastic gradient descent.

For computational speed-ups we expand the model with heuristics-based initialization of b_u, b_i and $\sum_{j\in N(u)} \mathbf{y}_j$ as used in [9], instead of learning these parameters. The parameters are initialized as follows:

$$b_i = \frac{\sum_{u\in N(i)} (r_{ui} - \mu)}{\lambda_1 + |N(i)|} \tag{9}$$

$$b_u = \frac{\sum_{i\in N(u)} (r_{ui} - \mu - b_i)}{\lambda_2 + |N(u)|} \tag{10}$$

$$\sum_{j\in N(u)} \mathbf{y}_j = \frac{\sum_{i\in N(u)} \mathbf{V}_i}{\lambda_3 + |N(u)|} \tag{11}$$

where μ is the sample mean over all ratings and $\mathbf{V}$ is the matrix of item factors computed from SVD of the observed rating matrix (with some imputing), and $\mathbf{V}_i$ are its rows.

F. Variational Autoencoder (VAE)

The VAE assumes a vector representation $\mathbf{x}_u$ for the ratings of user u. It consists of an encoder and a decoder neural network, which can be viewed as an inference model $q_\phi(\mathbf{z}_u|\mathbf{x}_u)$ and a generative model $p_\theta(\mathbf{x}_u|\mathbf{z}_u)$ respectively, where $\mathbf{z}_u \in \mathbb{R}^k$ is a latent representation of the user with a Gaussian prior distribution. The inference model is a variational distribution that approximates the true intractable posterior $p(\mathbf{z}_u|\mathbf{x}_u)$ and is set as a multivariate Gaussian with mean $\mu_\phi(\mathbf{x}_u)$ and variance $\text{diag}\{\sigma_\phi^2(\mathbf{x}_u)\}$[2]. In [5] the parameters are learned by maximizing a slightly modified ELBO function:

$$\mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{x}_u)}[p_\theta(\mathbf{x}_u|\mathbf{z}_u)] - \alpha \cdot \text{KL}(q_\phi(\mathbf{z}_u|\mathbf{x}_u)||p(\mathbf{z}_u)) \tag{12}$$

where the annealing factor $\alpha \in [0, 1]$ is intended to control the strength of regularization, allowing the model to better fit the data. Predictions are made by taking the output of the decoder network using the mean $\mu_\phi(\mathbf{x}_u)$ of the variational distribution as input.

[2]In order to be able to take the gradients with respect to ϕ during training of the network we perform the reparameterization trick. See [10].

III. Method

In this section we describe the setting and the workflow leading us to the final ensemble model.

A. Data & Metric

The data is extracted from a rating matrix over 10,000 users and 1000 items, where the ratings discretely scale from 1 to 5. There are 1,176,952 training samples with approximately mean 3.86 and variance 1.25, which occupy 11.77% of a full rating matrix. For each user or each item, the training dataset contains at least one observed rating. The held-out test set is of equal size to the training set. We note that higher rankings occur more frequently in the dataset. The aim here is to optimize root mean squared error (RMSE) on the test dataset. Given N samples, denote the real rating as r_i and its prediction as $\hat{r}_i$. The RMSE is calculated by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2}$$

B. Software & Hardware

All models were developed in Python3, with some Cython optimization for performance speed-ups. The models *A-E* were implemented using NumPy and Surprise[3] [11], the latter allowing for more convenient parameter tuning and data handling through the model selection and pre-processing functions it provides. Model *F* was implemented using PyTorch 1.4.0. Some of the models were optimized and trained on the ETH Euler cluster. The VAE was optimized and trained using one GTX 1080 GPU.

C. Model selection

For all models presented in Section II, whenever possible, model selection (i.e. of hyperparameters) was performed through k-fold cross validation and model performance was assessed on the public Kaggle test set.

The SVT algorithm, although possessing nice theoretical guarantees, was slow in practice as every step requires a computationally expensive SVD to be performed, making it computationally infeasible to tune its parameters.[4]

To account for the fact that some users tend to rate lower and some users tend to rate higher, the pLSA was improved by normalizing ratings through subtracting the user's rating average and scaling by the variance. We assumed that such an initialization would help the model learn the latent hidden states better. However, due to sparsity, good estimates of those statistics are hard to determine. We followed the strategy in [8] to calculate them with a smoothing strength of $q = 35$. Due to the size of the dataset, the EM-algorithm usually performs well with a smaller number of iterations. Lastly, an SVD reconstruction as in Section II-A was performed on the output from pLSA.

For SVD++, we extended the algorithm presented above to include learning rate decay and gradient momentum, in order to make the optimization numerically more stable.

Concerning the VAE, we adapted the approach taken in [5] where the user-item matrix is binarized for a classification setting aiming to optimize a ranking-based metric. Here we instead seek to minimize the RMSE and treat the problem as a regression task. A thorough hyperparameter selection was performed for the VAE model. We experimented with different values of the annealing factor, linearly increasing it from zero to a maximal value during the training. Furthermore, we regularized the model by adding dropout after each layer and by augmenting the input vector $\mathbf{x}_u$ with Gaussian noise during training. We selected hyperparameters such as the number of layers and bottleneck size using grid search, and used the Adam optimizer [14] with a fixed learning rate and gradient clipping.

D. Constructing an ensemble model

The models' predictions were clipped to the interval $[1.0, 5.0]$ to match the valid data range. In the experiments, we selected models with comparably competitive performance and averaged the clipped results to build an ensemble model for the final submission. Compared to each individual model, the ensemble averaging potentially has lower variance in RMSE (depending on the covariance of the base models) [6].

IV. Results

Comparing with [8], the results from grid search on the pLSA model suggest a smaller number of latent states z and EM iterations. Our best pLSA model uses $z = 20$ and 5 iterations. We then kept the eight largest singular values from SVD to reconstruct the rating matrix.

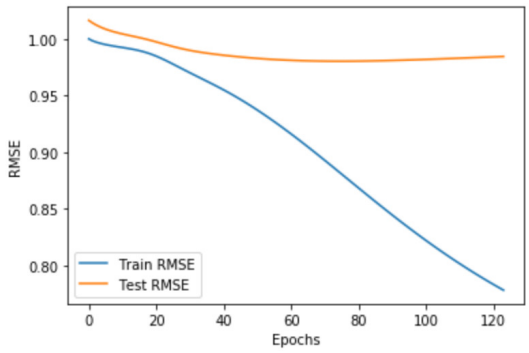

Figure 1. Training and test RMSE as a function of epochs for the SVD++

For SVD++, the grid search indicates a non-negligible difference in performance depending on choice of hyperparameters. As an illustration, Figure 1 shows the RMSE as a function of the number of epochs for a train-test-split of 0.25

[3]The implementation of SVD++ was also based on the one from Surprise.

[4]There do exist extensions to the algorithm aiming to decrease its run time, some of which were implemented without success. The curious reader is referred to [12] and [13].

(with all other hyperparameters taken from the final model). We observe that the test RMSE reaches its optimal value after around 80 epochs while the training RMSE continues to decrease, indicating a possible tendency of overfitting with too many epochs. Another noteworthy tuning decision is the imputation of matrix $\mathbf{V}$. Some experimentation gave an optimal imputed value of -5, which can be interpreted to combat the bias towards positive ratings among the observed.

Even more sensitive to hyperparameters is the VAE. Experimenting with different values of the annealing factor we find that omitting the KL term yields better performance, as can be seen in Figure 2. Furthermore, we show the impact of standard deviation of the Gaussian random noise augmentation in Figure 3. Introduction of noise helps to lower the validation error notably, and $\sigma = 0.5$ empirically gives the lowest validation RMSE.

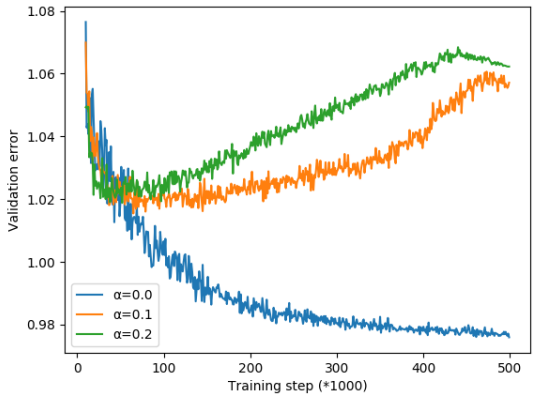

Figure 2. Experiments with annealing factor for the VAE

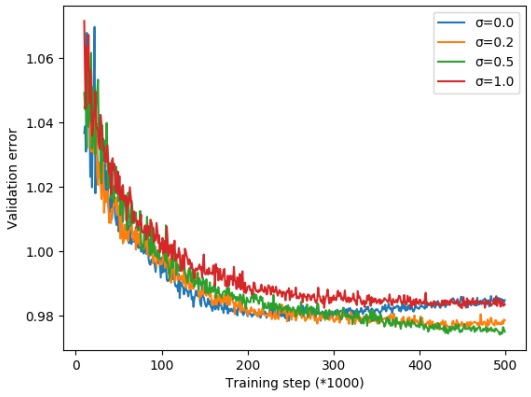

Figure 3. Experiments with added noise for the VAE

The RMSE scores on the public Kaggle test set of our models in II are presented in Table I. We see that SVD++ performs the best among the baseline models, followed by VAE and pLSA. We chose the three models for the ensemble as they performed significantly better than the rest and are deemed to be sufficiently diverse. The ensemble model produces a score of 0.97395 on the test set and thus performs favorably in comparison to the baselines, with only the SVD++ model being relatively close in performance.

Table I
MODEL PERFORMANCES ON THE KAGGLE PUBLIC TEST SET

Algorithm	RMSE
ALS	1.09192
SVT	1.02183
SVD	1.00432
pLSA	0.99011
VAE	0.98171
SVD++	0.97445
Ensemble	0.97395

V. DISCUSSION

Our data consists purely of co-occurrence observations of ratings, lacking meta-information such as item genre or timestamp. Within this context, the results from SVD++ and pLSA validate a strong predictive power of the low-rank representation retrieved via traditional techniques based on matrix factorization. In particular, the favorable results of the extended SVD++ show that initializing the bias terms and item factors via heuristics, instead of learning them by optimization, provides a good trade-off between computational efficiency and performance.

The VAE also shows great promise in performance compared to the baseline models. It requires considerable efforts for model selection however (in terms of architecture, regularization, optimization etc), only a subset of which could be explored within the scope of this paper. As shown in the experiments, its performance is highly sensitive to different configurations. Here, we highlight the influence of the annealing parameter. Our experiments show that the RMSE is considerably lower when the annealing is zero, consequently excluding the KL-divergence from the loss function. Thus with more time, we believe that the VAE could have been optimized even further, and perhaps an even more thorough experiment would produce results differing from ours. Future work can also be done to adapt other encoder-decoder structures, such as Transformers.

Lastly, we note the contribution made from constructing an ensemble of the models. The score of the ensemble averaging model greatly outperforms the average score of its contained base models, giving us an empirical indication of the ”wisdom of crowds” effect.

VI. SUMMARY

In this paper, we have worked on accurately recovering a rating matrix by an average ensemble model from SVD++, VAE and pLSA. The RMSE on the Kaggle public test set shows the strength of our model on unseen data. The extended SVD++ and pLSA models both contribute with high performance compared to baseline alternatives, as does the VAE after considerable efforts to tune its hyperparameters.

REFERENCES

[1] Y. Koren, "Factorization meets the neighborhood: a multi-faceted collaborative filtering model," in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. ACM Press, 2008.

[2] J.-F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on optimization*, 2010.

[3] S. Funk, "Netflix update: Try this at home," 2006. [Online]. Available: https://sifter.org/~simon/journal/20061211.html

[4] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*, 2018.

[5] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*. ACM Press, 2018.

[6] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Information Fusion*, 2005.

[7] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Algorithmic Aspects in Information and Management*, R. Fleischer and J. Xu, Eds. Springer Berlin Heidelberg, 2008.

[8] T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. Association for Computing Machinery, 2003.

[9] Z. Xian, Q. Li, G. Li, and L. Li, "New collaborative filtering algorithms based on svd++ and differential privacy," *Mathematical Problems in Engineering*, 2017.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[11] N. Hug, "Surprise, a Python library for recommender systems," http://surpriselib.com, 2017.

[12] J.-F. Cai and S. Osher, "Fast singular value thresholding without singular value decomposition," *Methods and Applications of Analysis*, 2013.

[13] T.-H. Oh, Y. Matsushita, Y.-W. Tai, and I. S. Kweon, "Fast randomized singular value thresholding for nuclear norm minimization," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.