doi: 10.36689/uhk/hed/2022-01-038

An Empirical Test of Stacked Autoencoder as Recommendation Model

Jaroslav LANGER*, Michal DOBROVOLNY and Ondrej KREJCAR

University of Hradec Kralove, Hradec Kralove, Czech Republic; jaroslav.langer@uhk.cz; michal.dobrovolny@uhk.cz; ondrej.krejcar@uhk.cz

* Corresponding author: jaroslav.langer@uhk.cz

Abstract: The Autoencoder method can be used for multiple scenarios, as it is very variable. In this case, the method is used for suggesting actions. This paper describes the theoretical aspects of the recommendation models. The next section describes the use of the autoencoder. There will be provided two experiments. The first one, recommendations for movie lens dataset with process of transformation datasets and proper model dimension setting included. The results of the first experiment are favorable. The second experiment for suggesting actions from custom workflow inspired dataset. In this experiment, two ways of preparing a dataset for a model will be tested. Unfortunately, results are worse than model example. The article concludes with a discussion of the results achieved in comparison with other authors where results with movie datasets.

Keywords: neural networks; autoencoder; recommender systems

JEL Classification: C45; C53; L86

1. Introduction

Referral systems are technologies and techniques that can provide recommendations for items to be used by a user. The recommendations provided are aimed at supporting their users in various decision-making processes, such as what products to buy, what music to listen to or which way to go. Accordingly, various techniques for generating recommendations in commercial environments have been designed and implemented. The aim of this research is to impose a certain degree of order on this diversity by presenting a coherent and unified repository of the most common methods of recommendation for solving the problem of collaboration filtering: from classical matrix factorization to high-end deep neural networks.

The field of computer learning is undoubtedly a big trend at the moment. Deep learning (DL) is a subset of computer learning that uses neural networks to analyze various factors with a structure that is similar to the human nervous system. One of the basic problems in this area is the quality of the input data, which directly affects the result of the model. Algorithms of DL can search and represent both structured and not structured data – for instance, natural language processing, time series or image data (Abdel-Nasser & Mahmoud, 2019; Pena-Barragan et al., 2011) In image data processing can be found examples about fixing an image (Wolterink et al., 2017; Yang, et al., 2018), compression (Sun et al., 2020), super-resolution (Dobrovolny et al., 2020; Christian, et al., 2017), image classification (Ciresan et al.,

2012; Mambou, et al., 2020), forecasting (Dobrovolný et al., 2020) session based recommendations. (Dobrovolny et al., 2020)

The article (Sedhain et al., 2015) describes the AutoRec model. The authors describe the implementation of the model and compare the results with a conventional matrix factorization algorithm. Furthermore, the authors focused on the influence of the choice of the activation function. Linear functions, namely ReLU and nonlinear Sigmoid, were compared. The ReLU activation function, which generally has better properties, showed worse results in this case than the Sigmoid function, which is often replaced by the ReLU activation function in modern networks. The next article (Kuchaiev & Ginsburg, 2017) focuses on the DeepRec model. The authors of the article describe the mentioned method and examine, apart from the influence of the activation function, especially the influence of the number of hidden layers on the results. Their results support the claim that deep learning with the use of an autoencoder can be used for recommendations even with a relatively small dataset.

Authors of article (Wu et al., 2016) are focused on Collaborative Denoising Autoencoders (CDAE). They assume with the given method that the available list of preferences is not complete and use a model to try to reconstruct it. It effectively uses a pattern of common preferences. The differences between linear and nonlinear activation function also enter into the whole observation.

Article (Liang et al., 2018) focuses on Multinomial Variational Autoencoders (MultVAE). Authors of the article introduce with multinomial probability using Bayesian statistics for parameter estimation. It introduces another regularization parameter, which has proven to be essential for achieving competitive performance.

One of other method is called Sequential Variational Autoencoders (SVAE). There is an article (Sachdeva et al., 2019) that focuses on it. The authors of the article extend the previously mentioned method. The authors transmit data, including a time sequence, within their model and show that the processing of time information is essential for improving the accuracy of the neural network result.

Another article (Steck, 2019), focuses on Embarrassingly Shallow Autoencoders (EAEC). The authors of the last model focus on the simplicity of the algorithm. They combine simple elements of a basic model to create a linear model that focuses on simple data. The results of some tested scenarios surpassed the far more complicated models described earlier in this work.

The paper's organization is structured as follows: In Sect. 2, we discuss the methods of content recommendation. Section 3 describes autoencoder model with and its results of two experiments. Finally, in Sect. 4, we conclude the paper and provide an outlook on our future work plans.

2. Methodology

At the core of any advanced proposal framework, which has seen enormous achievement in organizations like Amazon, Netflix and Spotify, is shared sifting. It works by social affair human decisions (known as appraisals) for items in a given area and coordinating people with similar requirements for information or similar tastes. Collaborative filtering system

clients share their insightful decisions and perspectives on everything they burn-through so other system clients can all the more likely evaluate which things to burn-through (Dobrovolny et al., 2020). Consequently, for new items, the collaborative filtering system offers helpful customized suggestions. In particular, the Autoencoder model will be discussed in this paper.

2.1. Autoencoders

Autoencoder is a type of neural network that can be used to learn a compressed representation of raw data.

An autoencoder is composed of an encoder and a decoder sub-model. The encoder compresses the input, and the decoder attempts to recreate the input from the compressed version provided by the encoder. After training, the encoder model is saved, and the decoder is discarded. (Hinton & Salakhutdinov, 2006; Liu et al., 2019; Vaiyapuri & Binbusayyis, 2020)

Usually, autoencoders consist of three-part: encoder – the part that includes input layer and hidden layer, bottleneck – this is where learned/compressed data is stored, and decoder – the part that starts from hidden layer and ends with output layer (Hinton & Salakhutdinov, 2006; Liu et al., 2019), as shown in following Figure 1.

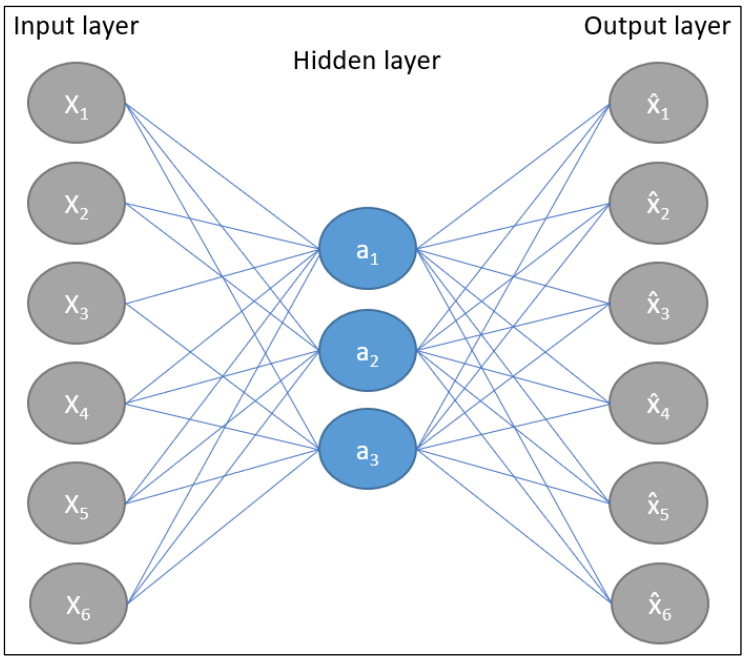

Figure 1. Autoencoder model schema

As visualized above, we can take an unlabeled dataset and frame it as a supervised learning problem tasked with outputting ($\hat{x}$), a reconstruction of the original input x.

This network can be trained by minimizing the reconstruction error, $(x, \hat{x})$, which measures the differences between our original input and the consequent reconstruction.

There are several main use cases for auto-encoders such as: Data compression, Denoising data, Classification and Collaborative Filtering (Recommendation). (Hinton & Salakhutdinov, 2006; Liu et al., 2019) In this paper, author will focus on the last one-use case

– Collaborative Filtering. There are many autoencoders model implementations. Some of them will be described and explained.

AutoRec In AutoRec, instead of explicitly embedding users/items into lowdimensional space, it uses the column/row of the interaction matrix as the input, then reconstructs the interaction matrix in the output layer.

On the other hand, AutoRec differs from a traditional autoencoder: rather than learning the hidden representations, AutoRec focuses on learning/reconstructing the output layer. It uses a partially observed interaction matrix as the input, aiming to reconstruct a completed rating matrix. In the meantime, the missing entries of the input are filled in the output layer via reconstruction for the purpose of recommendation. (Sedhain et al., 2015)

DeepRec DeepRec autoencoders extends idea of AutoRec. A deep autoencoder is composed of two, symmetrical deep-belief networks that typically have four or five shallow layers representing the encoding half of the net, and second set of four or five layers that make up the decoding half.

The layers are restricted Boltzmann machines (RBM), the building blocks of deep-belief networks.

A deep-belief network can be defined as a stack of restricted Boltzmann machines, in which each RBM layer communicates with both the previous and subsequent layers. The n nodes of any single layer don't communicate with each other laterally. (Kuchaiev & Ginsburg, 2017)

Collaborative Denoising Autoencoders (CDAE) is represented as a one hidden-layer neural network. CDAE consists of 3 layers, including the input layer, the hidden layer, and the output layer as is shown on next figure.

Collaborative Denoising Autoencoders extends idea of classic Denoising Autoencoders where key difference a latent vector for the user. In the input layer, there are in total I + 1 nodes, where each of the first I nodes corresponds to an item, and the last node is a user specific node (the red node in the figure), which means the node and its associated weights are unique for each user $u \in U$ in the data. (Wu et al., 2016)

Variational Autoencoders (VAE) Is special implementation of autoencoder. VAE provides a probabilistic manner for describing an observation in latent space. Instead of building an encoder which outputs a single value to describe each latent state attribute, it will formulate encoder to describe a probability distribution for each latent attribute.

Rather than directly outputting values for the latent state as it would in a standard autoencoder, the encoder model of a VAE will output parameters describing a distribution for each dimension in the latent space. Decoder model will then generate a latent vector by sampling from these defined distributions and proceed to develop a reconstruction of the original input.

2.2. Other Models

As other methods of collaborative filtering can be considered many different network architectures. Some of them are listed below.

Neural colaborative filtering Neural Factorization Machines for Sparse Predictive Analytics (He & Chua, 2017) is another parallel work that combines Factorisation Machines and Multi-Layer Perceptron seamlessly. This model brings together the efficacy of linear factorisation machines for sparse predictive analytics with the representation potential of nonlinear neural networks.

Bolzmann Machines Stochastic and generative neural networks capable of learning internal representations of difficult combinatorial problems are Boltzmann Machines. These models are commonly used for learning representations and solving them. These problems belong also to collaborative filtering. Boltzmann machines are two types – restricted Boltzmann machines (RBM) and explainable restricted Boltzmann machines (ERBM). RBM are specific by the structure. There are no output nodes. They only contain hidden and visible nodes. (Abdollahi & Nasraoui, 2016; Salakhutdinov et al., 2007)

Sequence modeling Simple but smart method for collaborative filtering seems to be sequence modelling. In paper Session-based recommendations using Recurrent neural networks - Long short-term memory (Dobrovolny et al., 2020) we introduced another method of modelling user inputs as sequences and learning them by word-level Long shortterm memory (LSTM). This method has lower accuracy but brings a possibility of real-time predictions on demand.

3. Using AutoEncoder as Recomendation Model

The neural network was created in Python using the PyTorch library. The model is a stacked autoencoder. A stacked autoencoder is a neural network consisting of several layers of sparse autoencoders, where the output of each hidden layer is connected to the input of a subsequent hidden layer. The author of the article Sparse, Stacked and Variational Autoencoder (Jonnalagadda, 2018) deals with this model.

Within the proposed model, it is possible to easily change some hyper parameters, such as learning speed, number of epochs, type of activation function, or type of optimizer. The Comet library is used to store the results from individual experiments.

3.1. Experiment 1 - Recomendation with Movielens Dataset

The first sample dataset to test the efficiency and accuracy of a neural network based on the Autoencoder method was obtained from a GroupLens (Harper & Konstan, 2015) project of the University of Minnesota Department of Informatics.

The data consists of 100,000 ratings of 1,682 movies in the format 1-5 from 943 users. Each user has rated at least 20 movies. In addition, basic information such as gender, age and nationality are available to users.

The model has a variable number of inputs and outputs depending on the dataset used as you can see in Figure 2.

Process of learning is described with following algorithm 1 below.

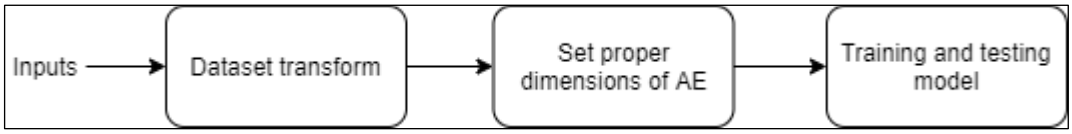

Figure 2. Flowchart of model processing

Algorithm 1: Prepare and train AE model

```
Result: list predictions
prepare datasets;
init modelwith proper dimensions and parameters;
init optimizer;
while epoch ¡ epochs count do
  make prediction;
  correct model using optimizer;
end
```

The network size with testing movie lens dataset is as follows:

```
SAE(
 (fc1):Linear(in_features=1682,out_features=20,bias=True)
 (fc2):Linear(in_features=20,out_features=10,bias=True)
 (fc3):Linear(in_features=10,out_features=20,bias=True)
 (fc4):Linear(in_features=20,out_features=1682,bias=True)
 (activation): Sigmoid()
)
```

Within the proposed model, it is possible to change some hyper-parameters and the result is monitored in the form of an error Mean Squared Error (MSE), which in this case indicates how much the final evaluation of the film by the user. For all experiments, a uniform number of epochs was chosen – 50 and the effects of the activation function and the optimizer were investigated.

Table 2. Measured values for different combinations of optimizers and activation functions

	Sigmoid	ReLU	Tanh
Adagrad	1.01909	1.03234	1.01908
Adadelta	1.00110	1.01802	1.01611
Adam	1.01772	1.01349	1.01775
RMSprop	1.00601	1.01704	1.01935

Table 2 shows that neither the activation function nor the type of optimizer used has a significant effect on the prediction results. In all cases, the error is around 1. This value indicates that the estimate of how the resulting film will like on a scale of 1-5 is with an error of 1. It can be said that the model is able to recommend films with high success based on previous evaluations.

3.2. Experiment 2 - Suggesting Actions with Workflow Inspired Dataset

The second experiment was focused on simulated workflow process. Simulated process is shown in Figure 3 below.

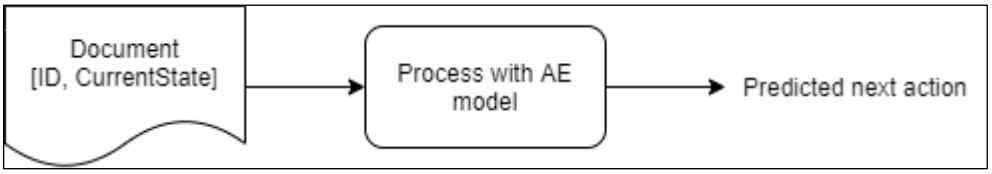

Figure 3. Flowchart of model processing

The aim of this experiment is to verify the ability to use an autoencoder for this type of use case. In document management software, the workflow for each document type would not have to be fixed and explicitly programmed for new customer types, but the model for predicting further action would learn the workflow itself based on a few examples.

Simulated dataset was about document's actions where part of the training dataset is shown below:

```
Document,currentState,nextAction
1,3,1
1,2,7
1,4,0...
```

This dataset consists of 50 records. Available actions and documents are shown below:

```
0::none
1::toManager
2::toAccounting
3::toSecretary
4::toControlling
5::toHR
6::toPlantManager
7::toPurchasing
8::toIT
```

```
1::invoice
2::priceOffer
3::order
4::employmentContract
5::ITguideline
6::EHSguideline
7::fine
```

Dataset had to be transformed to model's readable array in a form:

```
[ currAction | Doc1:nextAct | Doc2:nextAct | ... | Doc7:nextAct ]
```

MSE loss was 1.442. A better view of accuracy in this case will provide the number of successful predictions that was 36%.

There was the second attempt with following dataset transformation.

```
[ Document, CurrentAction, NextAction ]
```

MSE loss was 1.216. A better view of accuracy in this case will provide the number of successful predictions that was 42%.

Model was computed on desktop workstation computer with one dedicated graphic card GTX1070 operating with 8 GB Graphic memory supported with Intel Core i5 9600KF and 16 GB RAM. As a programming language we used Python in version 3.7

4. Discussion & Conclusions

Autoencoders has proven to be a good model for recommending movie content in terms of estimating user ratings based on selected parameters describing the person. In the first experiment in this article, it was managed to build on the results already achieved by other authors.

In the second experiment, which aimed to verify the usability of the autoencoder for predicting actions instead of explicit workflow programming, the model proved to be strongly unsatisfactory, despite two attempts with different dataset transformations. The conventional convolution forward network probably appears to be better for this type of use case, and the autoencoder did not surpass it in its properties, nor did it approach its results.

In an effort to find an equivalent in the process data with the structure of the movielens dataset, it occurred to us to use it for general recommendation of action on selected parameters, for example, persons who manipulated the document. This structure could be the subject of a possible further study.

Acknowledgments: The work was supported by the internal project "SPEV – Smart Solutions for Ubiquitous Computing Environments", 2022, University of Hradec Králové, Faculty of Informatics and Management, Czech Republic.

Conflict of interest: none

References

Abdel-Nasser, M., & Mahmoud, K. (2019, July). Accurate photovoltaic power forecasting models using deep LSTM-RNN. *Neural Computing & Applications, 31*(7), 2727–2740. https://doi.org/10.1007/s00521-017-3225-z

Abdollahi, B., & Nasraoui, O. (2016). *Explainable Restricted Boltzmann Machines for Collaborative Filtering*. http://arxiv.org/abs/1606.07129

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column Deep Neural Networks for Image Classification. In *2012 Ieee Conference on Computer Vision and Pattern Recognition* (pp. 3642-3649). New York: Ieee.

Dobrovolny, M., Krejcar, O., & Selamat, A. (n.d.). *Session based recommendations using Recurent neural networks - Long short-term memory.*

Dobrovolny, M., Mls, K., Krejcar, O., Mambou, S., & Selamat, A. (2020). Medical Image Data Upscaling with Generative Adversarial Networks. In I. Rojas, O. Valenzuela, F. Rojas, L. J. Herrera, & F. Ortuño (Eds.), *Bioinformatics and Biomedical Engineering* (pp. 739–749). Springer International Publishing. https://doi.org/10.1007/978-3-030-45385-5_66

Dobrovolný, M., Soukal, I., Lim, K., Selamat, A., & Krejcar, O. (2020). Forecasting of FOREX Price Trend using Recurrent Neural Network - Long short-term memory. In P. Maresova, P. Jedlicka, K. Firlej, & I. Soukal (Eds.), *Hradec Economic Days 2020* (pp. 95-103). Hradec Kralové. https://doi.org/10.36689/uhk/hed/2020-01-011

Harper, F., & Konstan, J. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems, 5*(4), 1-19. https://doi.org/10.1145/2827872

He, X., & Chua, T. S. (2017). Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR '17: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 355-364). Shinjuku Tokyo Japan: Association for Computing Machinery.

Hinton, G., & Salakhutdinov, R. (2006, July 28). Reducing the dimensionality of data with neural networks. *Science, 313*, 504-507.

Jonnalagadda, V. (2018, December 06). *Sparse, Stacked and Variational Autoencoder*. https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64

Kuchaiev, O., & Ginsburg, B. (2017, October 10). Training Deep AutoEncoders for Collaborative Filtering. *arXiv*. http://arxiv.org/abs/1708.01715

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2016). Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *30th Ieee Conference on Computer Vision and Pattern Recognition* (pp. 105-114). New York: Ieee. https://doi.org/10.48550/ARXIV.1609.04802

Liang, D., Krishnan, R., Hoffman, M., & Jebara, T. (2018, February 15). Variational Autoencoders for Collaborative Filtering. *arXiv*. http://arxiv.org/abs/1802.05814

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. (2019). A survey of deep neural network architectures and their applications. *Neurocomputing, 234*, 11-26. https://doi.org/10.1016/j.neucom.2016.12.038

Mambou, S., Krejcar, O., Selamat, A., Dobrovolný, M., Maresova, P., & Kuca, K. (2020). Novel Thermal Image Classification Based on Techniques Derived from Mathematical Morphology: Case of Breast Cancer. In I. Rojas, O. Valenzuela, F. Roja, L. Herrera, & F. Ortuño (Eds.), *Bioinformatics and Biomedical Engineering* (Vol. 12108, pp. 683-694). Springer International Publishing. https://doi.org/10.1007/978-3-030-45385-5_61

Pena-Barragan, J. M., Ngugi, M. K., Plant, R. E., & Six, J. (2011). Object-based crop identification using multiple vegetation indices, textural features and crop phenology. *Remote Sensing of Environment, 115*(6), 1301-1316. https://doi.org/10.1016/j.rse.2011.01.009

Sachdeva, N., Manco, G., Ritacco, E., & Pudi, V. (2019). Sequential Variational Autoencoders for Collaborative Filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (pp. 600-608). Melbourne: Association for Computing Machinery. https://doi.org/10.1145/3289600.3291007

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning - ICML '07* (pp. 791-798). Corvalis, Oregon: ACM Press. https://doi.org/10.1145/1273496.1273596

Sedhain, S., Menon , A., Sanner, S., & Xie, L. (2015). AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 111-112). Florence: Association for Computing Machinery. https://doi.org/10.1145/2740908.2742726

Steck, H. (2019). Embarrassingly Shallow Autoencoders for Sparse Data. *The World Wide Web Conference on - WWW'19*, 3251-3257. Retrieved October 28, 2020, from http://arxiv.org/abs/1905.03375

Sun, Y., Jiwei, C., Liu, Q., & Liu, G. (2020). Learning image compressed sensing with sub-pixel convolutional generative adversarial network. *Pattern Recognition, 98*, 107051. https://doi.org/10.1016/j.patcog.2019.107051

Vaiyapuri, T., & Binbusayyis, A. (2020). Application of deep autoencoder as an one-class classifier for unsupervised network intrusion detection: a comparative evaluation. *Peerj Computer Science*, 327. https://doi.org/10.7717/peerj-cs.327

Wolterink, J., Leiner, T., Viergever, M. A., & Išgum, I. (2017). Generative Adversarial Networks for Noise Reduction in Low-Dose CT. *IEEE Transactions on Medical Imaging, 36*(12), 2536-2545. https://doi.org/10.1109/TMI.2017.2708987

Wu, Y., DuBois, C., Zjeng, A. X., & Ester, M. (2016). Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining* (pp. 153-162). San Francisco: Association for Computing Machinery. https://doi.org/10.1145/2835776.2835837

Yang, Q., Yan, P., Zhang, Y., Yu, H., Shi, Y., Mou, X., Kalra, M. K., Zhang, Y., Sun, L., & Wang, G. (2018). Low-Dose CT Image Denoising Using a Generative Adversarial Network With Wasserstein Distance and Perceptual Loss. *Ieee Transactions on Medical Imaging, 37*(6), 1348-1357. https://doi.org/10.1109/TMI.2018.2827462