ARTICLE IN PRESS
JID: PATREC [m5G;August 4, 2018;16:6]
Pattern Recognition Letters 000 (2018) 1–9
Contents lists available at ScienceDirect
Pattern Recognition Letters
journal homepage: www.elsevier.com/locate/patrec

Rating prediction via generative convolutional neural networks based regression

Xiaodong Ning*, Lina Yac, Xianzhi Wang, Boualem Benatallah, Manqing Dong, Shuai Zhang

University of New South Wales, k 17 cse building, Sydney, Australia

ARTICLE INFO

Article history:
Available online xxx

MSC:
41A05
41A10
65D05
65D17

Keywords:
Generative convolutional neural network
Rating Prediction

ABSTRACT

Ratings are an essential criterion for evaluating the quality of movies and a critical indicator of whether a customer would watch a movie. Therefore, an important related research challenge is to predict the rating of a movie before it is released in cinema or even before it is produced. Many existing approaches fail to address this challenge because they predict movie ratings based on post-production factors such as review comments from social media. Consequently, they are generally inapplicable until a movie has been released for a certain period of time when a sufficient number of review comments have become available. In this paper, we propose a regression model based on generative convolutional neural networks for movie rating prediction. Instead of post-production factors widely used by previous work, this model learns from movies' intrinsic pillars such as genres, budget, cast, director and plot information, which are obtainable before the production of movies. In particular, the model explores the correlations between the rating of a movie and its intrinsic attributes to predict its rating. The results can serve as a reference for investors and movie studios to determine an optimal portfolio for movie production and a guidance to the interested users to choose the movie to watch. Extensive experiments on a real dataset are benchmarked against a set of baselines and state of the art approaches. The results demonstrate the effectiveness of our approach. The proposed model is also general to be extended to handle other prediction tasks.

1. Introduction

Movies have been an important source for daily recreation in the modern society and have gained increasing popularity in recent years due to the introduction of wider screen and iMAX. Nowadays, more than 4,000 movies are produced globally each year and America itself produces over 700 movies annually.[1] These movies are widely discussed on social media and rated on various movie-rating websites. Those ratings are indicative of many important performance metrics of movies, such as the prospective sizes of the audience and potential box office revenue of the movies.[2] This makes it important to predict the ratings of movies to foresee their performances and to adjust their production and propagation plans.

Many existing approaches for object (e.g., movie) rating prediction are specific to a single user [2,4,10,20,23–25]. Thus, they are unsuitable for predicting the ratings of a movie from the public.

Some other research focus on qualitative assessment of each movie instead of predicting a specific rating. For example, Tan et al. [19] classify the movies into three categories:good, average, and bad depending on its rating in the Internet Movie Database (IMDB). The authors try to utilize the movie's structure to predict the final performance of movie. Their experimental results show it indeed exist a correlation between movie structure and movie perceived rating. Kabinsingha et al. [8] classify the movies into:PG, PG-13 and R. The authors utilize various attributes of movies to predict the movie rating categories. However, these qualitative assessment can not provide accurate and meaningful ratings of movies.

In view of this deficiency, we focus on predicting the cohort review ratings for the movies. An effective cohort rating prediction not only provides people with an intuition of how a movie is worth watching but also helps movie makers make wise decisions on choosing the most promising combination of resources such as directors, actors, and movie plots for the movie production. Generally, the earlier we can predict the rating of a movie, the more valuable information can be provided to guide the

* Corresponding author.
E-mail addresses: xiaodong.ning@ad.unsw.edu.au, z5122770@student.unsw.edu.au (X. Ning).
[1] http://www.mpaa.org/wp-content/uploads/2016/04/MPAA-Theatrical-Market-Statistics-2015_Final.pdf.
[2] http://collider.com/box-office-reviews-statistics/.

https://doi.org/10.1016/j.patrec.2018.07.028
0167-8655/© 2018 Elsevier B.V. All rights reserved.

movie makers. Although there are several previous research efforts [12,16,18,21] on cohort rating prediction for movies, products, or services, most of them predict movies rating based on the word of mouth(WOM), such as the tweets contents, review comments of movies on websites(IMDB, Rotten Tomatoes, etc). Some typical examples include: Liu et al. [12] develop a movie-rating system which predicts movie rating based on the sentiment of review-summarization. They collected the movie reviews from Internet Blogs that do not consist of any rating information. Sentiment analysis is performed to determine the semantic orientation of the reviews and movie-rating score is based on the sentiment-analysis result.; Wijaya and Bressan [21] adopt the idea of random walk using PageRank to rank movies according to the opinions expressed in their online textual reviews. In particular, they first construct a sentiment graph from the collocation of adjectives, followed by computing the semantic orientation scores using PageRank algorithm; finally, they aggregate the semantic orientation scores of adjectives in all the reviews of a movie to compute the final rating. Schmit and Wubben [18] also tries to predict movie ratings using the content of tweets. They apply unigrams, bigrams, trigrams and combinations of these n-grams of twitter content and then transform them into TF-IDF vectors which is the feature. Then they apply different machine learning algorithms for rating prediction. Similar to Schmit and Wubben [18], Oghina et al. [16] predict IMDb movie ratings and consider two sets of features: surface and textual features. The surface feature is gained by capturing the amount of activity around a movie title which is called as quantitative indicators. For the textual features, they assume that no social media signal is isolated and use data from multiple channels that are linked to a particular movie, such as tweets from Twitter and comments from YouTube. The experimental results show their work is able to rate movies very close to the observed values. Different from the above work, Navarathna et al. [13] proposes a novel method of representing audience behavior through facial and body motions from a single video stream. The method then uses these features to predict the rating for feature-length movies. The limitation of methods of this category is that, both the review comments and audience behavior used in the above research are only available after the movies have been released for a certain period of time. It does not make sense for movie producers to predict movie rating after the release of a movie. The most similar worker to ours is proposed by Hsu et al. [5], who proposes to predict movie ratings with IMDB attributes. However, the training samples used in the experiments are much larger in number than the testing samples (the ratio of the two types of samples is approximately 22.5:1). This makes the experimental results not convincing enough.

In view of the deficiencies of existing efforts, we propose a regression approach based on generative convolutional neural networks for predicting movie ratings at early stages of movie production. The predicted results can therefore be used to guide movie makers on their movie production plans, as well as to guide the potential audience to choose the high-quality and welcomed movies. Our model consists of three main components: artificial feature generator, artificial label generator, and discriminator. The former two generators create new samples to make up for the likely case of lack of training samples while the discriminator is a one-dimensional convolutional neural network which captures the complex relationships between different features to make accurate rating predictions. Our contributions are listed as follows:

- We design a set of high-quality intrinsic features to characterize various aspects of a movie such as genres, budget, release date, historical information of cast and director, and plot topics. All these aspects of information can be extracted at a very early stage before a movie is released.
- We propose an effective regression model based on generative convolutional neural networks for predicting movie ratings. Our model also includes a solution to the problem of insufficient training samples.
- The experimental results on a real IMDB dataset show that our model outperforms a series of baselines and state of the art methods with an improvement of 9.3% in MSE and 7.6% in Hit ratio, respectively. Furthermore, we also conduct ablation study which confirms the effectiveness of the components in our model.

The rest of this paper is organized as follows. Section 2 presents the methodology of our approach. Section 3 details the experimental settings, baseline methods and comparison results. Finally, conclusion and future are presented in Section 4.

2. Methodology

In this Section, we illustrate the feature extraction (Section 2.1), detail the feature transformation process (Section 2.2), and finally, present our model structure (Section 2.3).

2.1. Feature extraction

There are various feature selection methods in previous work [26,27]. However, they are not very suitable in out case. In this context, we extract several types of features for movie rating prediction:

2.1.1. Numerical features

According to the previous study[3], the popularity of the director and actors can play an important role in the final rating of a movie. Thus we extract the Facebook likes of the director, top three leading actors (here we choose three as we assume the three leading actors can influence the quality of movie) and the cast of each movie in our dataset. Additionally, we assume the history rating and box office of director and top three leading actors can significantly influence the final rating of a movie. In this context, we extract the history ratings and box office for the director and leading actors of the current movie from the previous movies they participated in. Then, we calculate the average historical ratings and box office of director and leading actors for the current movie. Besides the above features, the budget is another dominant factor in a movie's quality, which can, in turn, influence the final rating. So we add it as another feature. Additionally, we also take the duration of each movie into account. In case of the missing values, we fill them with the median values of their respective feature. Totally, the numerical features add up to a 15-dimensional vector.

2.1.2. Categorical features

In preparation of the categorical features, we list all the variables that appear in each feature in the dataset and represent the occurrence of each variable in a bitmap. Previous studies[4] show that genres of movies are associated closely with their final ratings. Therefore, we collect total 21 genres occurring in the movies including 'Sci-Fi', 'Crime', 'Comedy', 'Thriller', 'Drama', etc. For each genre feature, we assign 1 to the genre to which the movie belongs and otherwise 0. It is notable that one movie can belong to several genres. The MPAA rating designated to classify movies with regard to suitability for audiences in terms of issues such as sex, violence, substance abuse, profanity, impudence or other types of mature content. It is also assumed to be a useful index for the final

[3] https://nycdatascience.com/blog/student-works/web-scraping/movie-rating-prediction/.
[4] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0136083.

movie rating. Therefore, we also take the MPAA rating of movies into account. There are totally five categories of MPAA ratings: R, PG-13, PG, G and NC-17. We transform the five categories into 1-5 categorical features. One movie can belong to only one MPAA rating category. Specially, we also add the release date as a feature and split it into a three-dimensional vector composed by weekday (we transform the day into specific weekday which is meaningful), month and year. In this section, we get a 29-dimensional vector totally.

2.1.3. Topical features

We use the 'Bag of words' representation, which is commonly used in natural language processing, to characterize movie plots. The plot description of each movie generally contains 200 to 300 words for the IMDB dataset. We first eliminate the stop words, meaningless words, and numbers, and then perform stemming (here we utilize PorterStemmer[5]) to transform each word to its simplest form. Finally, we construct a word-document matrix by tabulating the words which occur in one or more movies plots. The importance of each word is defined by a TF-IDF weighting scheme as follows:

$$I_i = \frac{d_i}{D} \times N_i$$

where d_i is the number of movie plots that contain the ith word, D denotes the total number of movies, and N_i is the total times of occurrence of the ith word across all the movies. It is notable that To eliminate the low important words, we keep only the top 5,000 most important words in a data-driven manner and perform the latent Dirichlet allocation (LDA) to extract the latent topics from movies. We set the topic number to its empirically optimal value of 25.

Finally, we get a 69-dimensional feature vector.

2.2. Feature transformation

After the above steps, we get a 69-dimensional feature vector. As the values of each feature may fall in significantly different ranges, we normalize each feature by the following equation:

$$Nomfeature_i(j) = \frac{Feature_i(j)}{(\mathrm{Max}(Feature_i) - \mathrm{Min}(Feature_i))} \tag{1}$$

where $Nomfeature_i(j)$ denotes the normalized the ith feature from the jth sample; $Feature_i(j)$ denote the original feature; Max ($Feature_i$) denotes the maximum ith feature value from all the samples; Min ($Feature_i$) denotes the minimum ith feature value from all the samples;

After normalization, the values of all the features fall into range *0-1*. Then we feed the normalized feature into an unsupervised auto-encoder network, which we use to reduce the dimensions of feature and capture the complex relationships between features. We choose the commonly used base auto-encoder structure: one input layer, one hidden layer, and one output layer. The hidden layer learns to encode the input feature vector whilst the output layer learns to decode it and the hidden layer produces the final feature vector. Suppose X is the input feature vector and X' is the reconstructed feature vector. The dimensions of them is the same. The process goes as follows:

$$X' = Decode(Encode(X)) \tag{2}$$

The objective of auto-encoder(AE) is to minimize the reconstruction error between the input value X and the reconstructed value X'.

$$L = \mathrm{Min}||(X - X')||_p \tag{3}$$

In our AE model, we set the dimensions of the hidden layer as 50, learning rate as 0.005, training epochs as 1500. After the feature transformation, we finally get a 50-dimensional feature vector.

2.3. Model learning

Deep learning has been proven to achieve promising performance in various domains such as image processing [11] and speech recognition [22], thanks to its ability to model high-level representations and capture complex relationships of data via a stacking multi-layered architecture. Among the deep learning techniques, Convolutional Neural Network(CNN) has been recently extended to combine the complex relationships of different kinds of features in many other areas [14,15]. Owing to the capability of CNN, we decide to utilize the one dimensional convolutional neural network(1D-CNN) in our model as our feature is one dimensional vector. Generally speaking, the performance of deep learning technique significantly relies on the number of training samples. In other words, increasing the number of training samples is an effective method to improve the final performance. As in our work, the total amount of movies from IMDB dataset is not very large while some of them lack important statistical information which makes matter worse. In this context, the available training samples are limited for training an accurate prediction network. In order to increase the number of samples, we propose a regression model based on generative convolutional neural networks to generate artificial samples and to combine them with real samples to co-train the entire network. The overall structure is shown in Fig. 1. From the figure, we can find that our model consists of three components: artificial feature generator(noted as G_1), artificial label generator(noted as G_2) and discriminator. We introduce the three components as follows.

2.3.1. Artificial feature generator G_1

The artificial feature generator G_1 has two layers and is used to generate the artificial features. Firstly, we initialize *Nseed* seed vectors with *Lseed* length. Each point in the seed vector is generated randomly from range 0-1. Then, the seed vector is fed into the two fully-connection layers (the neurons in each layer are $N1$ and 50). We apply Relu (Rectified Linear Units) activation function on the dot product results generated from each layer. Then we take the first fully-connection layer as an example to show the details of the fully-connected layer and relu function in the equation. V_s denotes the seed vector; $W1$ denotes the weight matrix with shape ($N1 \times Lseed$); $B1$ denotes the bias vector with $N1$ length; *Output*($N1$) denotes the output vector of this layer.

$$Output(N1) = W1 \times V_s + B1 \tag{4}$$

Via the two fully connected layers, the random seed vector will be transformed into a 50-dimensional vector (the length is same as the real feature vector).

2.3.2. Artificial label generator G_2

As the generated artificial features from G_1 require ratings, we design an artificial label generator G_2 to generate the corresponding ratings for them. As the artificial feature vector and real feature vector share the same dimensions, we decide to apply the K nearest neighbors regression to label the artificial samples using the real training samples. The distance between artificial samples and real samples are calculated in euclidean distance. We describe the details in the Eq. (5), where D_{ij} denotes the distance between *ith* artificial sample and *jth* real sample; N denotes the length of feature(50 in our work); F_{ik} denotes *kth* feature value of *ith* artificial

[5] http://www.nltk.org/howto/stem.html.

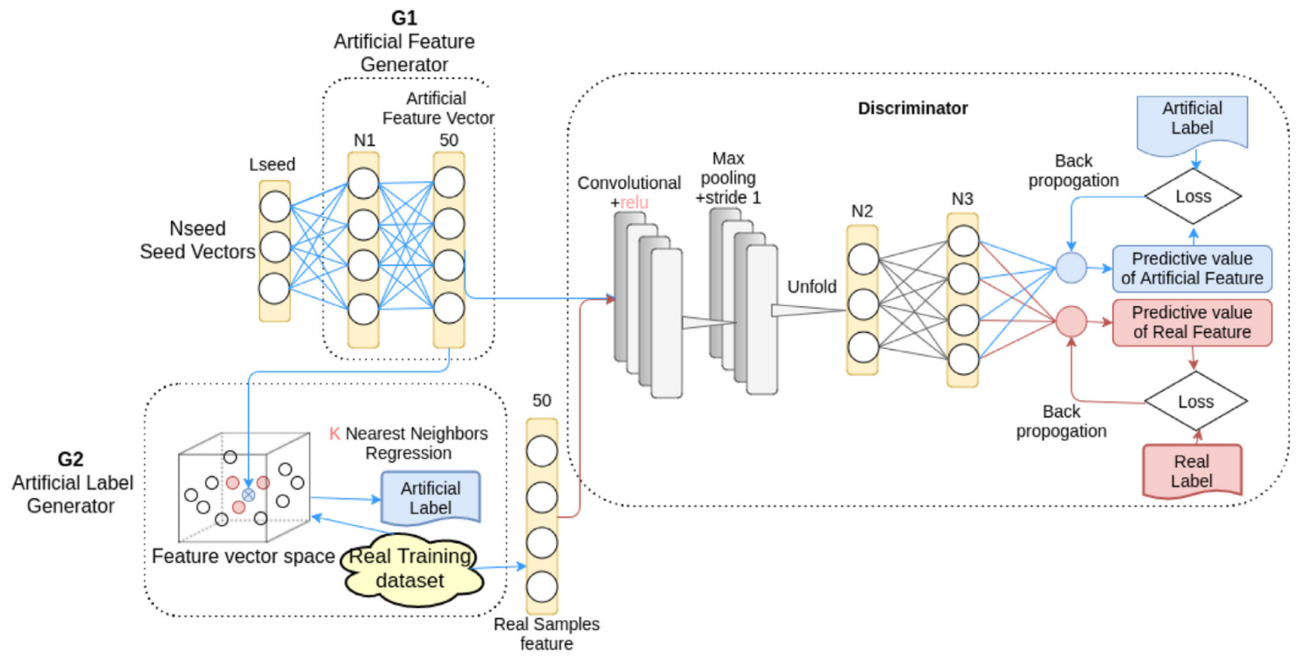

Fig. 1. Generative convolutional neural networks based regression model.

sample; F_{jk} denotes kth feature value of jth real sample;

$$D_{ij} = \sum_{k=1}^{N} \frac{(F_{ik} - F_{jk}) * 2}{N} \tag{5}$$

After calculating the distances, we apply KNN regression to calculate the ratings for artificial samples. We describe the details of the process in Eq. (6), where $Artificialrating_i$ denotes the generated rating of the ith artificial feature; K denotes the top K nearest real features to the ith artificial feature; D_{ij} denotes the distance between ith artificial sample and jth nearest real sample; R_j denotes the rating of jth nearest real sample

$$Artificialrating_i = \frac{\sum_{j=1}^{K} D_{ij} * R_j}{\sum_{j=1}^{K} D_{ij}} \tag{6}$$

2.3.3. Discriminator

Via the generator G_1 and G_2, we get a series of artificial samples with feature and label. After generating the artificial samples, we then feed them with the real samples into the discriminator. The discriminator is actually a one-dimensional CNN based regression network that consists of four layers: one convolutional layer, a max pooling layer, and two fully-connected layers. The convolutional layer in discriminator takes a set of Fn independent filters and slides them over the whole feature vector with stride size Fs and filter length Fl. Along the way, the dot product is taken between the filters and chunks of the input feature. Then we apply Relu (Leaky Rectified Linear Units) activation function on the dot product results generated from filters to add the nonlinear relationship into the network and help to alleviate the vanishing gradient problem. Filters are used to generate the feature vectors in each filter length. The same padding is used by the convolutional layer to keep the output remain the same size as the input. In this way, the original feature vector is projected into a stack of feature maps (vector maps in our work). We show the details convolutional and relu function in Eqs. (7) and (8) respectively, where $Conv_i(j)$ denotes the jth convolutional output from the ith filter and input vector I (50 dimensional vector in our discriminator); F_i denotes the ith filter vector; $Convrelu_i(j)$ denotes the relu function result of the $Conv_i(j)$.

$$Conv_i(j) = I[j * Fs, (j * Fs + Fl)] * F_i \tag{7}$$

$$Convrelu_i(j) = \mathrm{Max}(0, Conv_i(j)) \tag{8}$$

The multiple filters in convolutional layers can be updated automatically via the evolution of the network. Following the convolutional layer, we add one max-pooling layer with max-filter length Pl and stride size Ps. Similar to previous part, we show the details of max-pooling layer in Eq. (9), where Max (i, j) denotes the jth max-pooling output from the ith convolutional layer output with Relu $Convrelu_i$.

$$\mathrm{Max}(i, j) = \mathrm{Max}(Convrelu_i[j * Ps, (j * Ps + Pl)]) \tag{9}$$

The convolved and max-pooled feature vectors will be unfolded and fed into two fully-connected layers (the neurons in each layer are $N3$ and 1) with dropout probability (Kp) applied for the high-level reasoning.

Neurons in each fully-connected layer have full connections to all activations in the previous layer in regular Neural Networks. Their activations can, therefore, be computed using matrix multiplication followed by a bias offset. Finally, the initial feature vector is transformed and projected into one dimensional value via the second fully connected layer. The one-dimensional value generated from the final layer is the predictive rating.

The objective of our model is to predict the label as a continuous value instead of a class, so the commonly used cross entropy loss function should be modified. As there are two types of samples(artificial samples and real samples) in our model, we define two loss functions used in our model as follows:

$$\mathcal{L}_1 = \mu * \frac{\sum_{i=1}^{K_1} (A_{pi} - A_{ti})^2}{K_1} \tag{10}$$

$$\mathcal{L}_2 = \frac{\sum_{i=1}^{K_2} \left(R_{pi} - R_{ti}\right)^2}{K_2} \tag{11}$$

where $\mathcal{L}_1$ denotes the first loss function from artificial samples; $\mathcal{L}_2$ denotes the second loss function from real samples; R_{pi} denotes the predictive label of ith real sample; R_{ti} denotes the true label of *ith* real sample; K_1 and K_2 denotes the total number of real samples and artificial samples respectively; A_{pi} denotes the predictive label of *ith* artificial sample; A_{ti} denotes the artificial labels of *ith* artificial sample; μ(0-1) tunes the weight of $\mathcal{L}_1$ balanced with $\mathcal{L}_2$ during the training process

2.3.4. Network training

The artificial feature generator G_1 and discriminator are trained via back-propagation while the artificial label generator G_2 requires no training at all. Adaptive Moment Estimation (Adam) optimizer [9] is used in the training process which is a method that computes adaptive learning rates for each parameter.

Adam optimizer is demonstrated empirically to show that convergence meets the good performance fast especially in CNN based network.

As we have two loss functions($\mathcal{L}_1$ and $\mathcal{L}_2$) from real samples and artificial samples, the discriminator is trained iteratively by them to achieve a satisfactory prediction performance. By doing so, we can augment the available training samples to train a better discriminator. The weight μ in loss function $\mathcal{L}_1$ is used to balance the impact of artificial samples on the discriminator training process. In comparison, the feature generator G_1 is trained only with artificial samples($\mathcal{L}_1$). The G_1 generator evolves subsequently following the discriminator using $\mathcal{L}_1$ and updates its parameters via back-propagation to produce highly realistic artificial features. It is notable that the evolution of generator G_1 will also improve the training effect of the discriminator.

Via the optimization process of two loss functions, feature generator G_1 will learn how to generate high-quality artificial samples which is highly similar to the real samples, and discriminator will learn to predict the final labels accurately.

3. Experiments

In this section, we will describe the dataset in Section 3.1, and detail the baseline methods and their parameters settings in Section 3.2. Finally, the comparison results are shown in Section 3.3.

3.1. Settings

Dataset. We collect the movie attributes from the IMDB dataset in Kaggle and crawl the plot information from the IMDB website. Totally, there are 5,004 movies, from which we only keep movies with plot descriptions produced in the USA and get 2,571 movies. Each movie record contains attributes including facebook likes of directors and cast, genres, country, MPAA rating,[6] release date, budgets of the movie and box office records of directors and actors. As we require the adequate history ratings and box office of directors and actors, we only keep the 1471 movies released after 2003. It is notable that all the historical features of samples in our dataset are only extracted from the previous movies whose release dates are earlier than the samples. By doing so, we can avoid using any ‘future information’ in our dataset. Finally, we split the movies into two part: training samples(1219 movies released between 2003-2013) and testing samples(252 movies released after 2013). The distribution of rating scores of all the dataset is shown in Fig. 2. In order to avoid over-fitting in the testing samples, we utilize five-fold cross-validation training of our model in training samples.

[6] https://en.wikipedia.org/wiki/Motion_Picture_Association_of_America_film_rating_system# MPAA_film_ratings.

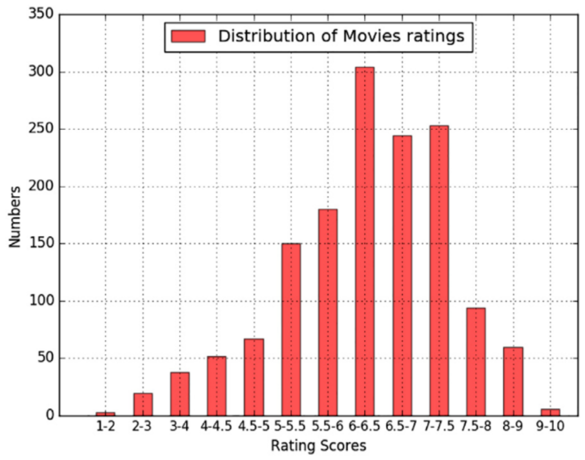

Fig. 2. Score distribution of dataset.

3.2. Baseline methods

In this paper, we choose the following baseline methods and tune their parameters to optimize the performance.

- *Decision Tree Regression* is a regression version of decision tree, here we set max depth as 4;
- *Random Forest Regressor* is a regression version of random forest, here we set 200 estimators and max depth as 5
- *Gradientboosting* is a prediction model in the form of an ensemble of weak prediction models(decision tree), here we set 50 estimators, learning rate as 0.1, random state as None.
- *Adaptive Boosting(Adaboosting)* with base estimator as decision tree regressor, 50 estimators, learning rate as 0.5, loss function as square, random state as None.
- *Extreme Gradient Boosting(Xgboosting)* with 100 estimators, learning rate as 0.8, subsample as 0.75, max depth as 5.
- *Support Vector Regression* [1] The idea of SVR is based on the computation of a linear regression function in a high dimensional feature space where the input data are mapped via a nonlinear function, here we set penalty parameter as 0.8, epsilon as 0.2, kernel function as ‘rbf’(radial basis function).
- *KNN-based Regression method* [17], where the average ratings of the top K nearest neighbors are computed as the predictive value for each movie according to the clustering results of KNN. Specifically, we set K=7 as the neighbor number for the KNN-based method;
- *Hypergraph Regression model* is a modified regression version of the hypergraph classification method [7], where we define a hyperedge by each movie and its 10 nearest neighbors to form a hypergraph. The weight of each hyperedge is calculated using the mean similarities of pairs in this hyperedge. Each movie rating is calculated by the following equation:

$$R_i = \sum_{j=1}^{N_j} H_j \sum_{k=1}^{10} S_{ik} * R_k$$

where R_i is the predicted rating of the ith movie, N_j is the number of hyperedges the ith movie belonging to, k ranges from 1 to 10 and denotes the movies in the same jth hyperedge with movie i, H_j denotes the corresponding hyperedge weight, S_{ik} denotes the similarities between the target movie and its neighbors, and R_k denotes the rating of movie’s kth nearest neighbor;
- *Kernel-1 Regression method* is a kernel-based approach for movie box office prediction used in [3]. We borrow this approach in our case and use the parameter setting recommended in the paper. This approach can be regarded as an improved version

of KNN regression which consists of three steps. The first step is to measure the distances between the new sample with all samples in training dataset. Different from KNN, the kernel approach calculates a weighted Euclidean distance between two feature vectors, which assign a 'feature weight' V_j for the *jth* feature in the feature vector. The second step is to normalize each distance by all the pairwise distances with a scaling parameter θ. Both the 'feature weight' vector $\vec{V}$ and the scaling parameter θ are randomly initialized and calibrated via minimizing the final loss function.

- *Deep Belief Network Regression* A deep belief network is proposed by Hinton where deep belief network is placed at the bottom for unsupervised feature learning with a linear regression layer at the top of supervised prediction. We set three hidden layers (with 110,200,330 units in each layer).
- *Multiple Layer Perception(MLP)* with three layers (100,300,1 neurons in each layer), number of iterations as 1200.
- *One Dimensional Convolutional Neural Network(1DCNN)* [11] with three layers: one convolutional layer(number of filters as 4, filter length as 3, stride as 1), two fully-connected layer (300 and 1 neurons in them), learning rate as 0.003.
- *Long short term memory neural network(LSTM)* with three layers(input layer, LSTM layer, output layer). The forget bias(controls the percentage of the cell 'memory') is set as 0.7, number of neurons in each layer is set as 40.
- *Long short term memory neural network based linear regression(LSTM-LR)* with linear regression as the final output layer. It utilizes the feature extracted from LSTM as input of linear regression. The parameters in LSTM layer is the same as the LSTM.
- *One Dimensional Convolutional Neural Network with Support Vector Regressor(1DCNN-SVR)* is the regression version of Huang's work [6] which utilizes the support vector regressor to replace the output layer of 1DCNN. The parameters are set same as the 1DCNN with support vector regressor.

Parameter tuning. We conducted extensive experiments to determine the optimal configuration of parameters for our model. There are two types of parameters in our model: the first type includes weights and biases in the model layers, which can be initiated randomly and learned afterwards from each iteration; the second type includes the parameters that should be configured manually. In particular, we select 12 most common hyper-parameters for our model, namely the number of seed vectors in feature generator(G_1) *Nseed*, the length of the seed vector *Lseed*, the number of neurons in the first fully connected layer in G_1 *N*1, the number of nearest neighbors of the generated feature in label generator G_2 *K*; number of independent filters in convolutional layer of discriminator *Fn*, the length of filter *Fl*, stride of filter *Fs*, max-filter length *Pl* and stride size *Ps* in max pooling layer of discriminator, the number of neurons of first fully connected layer in discriminator *N*3, dropout probability *Kp*, learning rate *lr*, the weight between two loss functions μ. Since the weight *W* and bias *b* in each neural layer can be learned automatically via the evolution of model network, we focus on tuning the hyper-parameter: *Nseed, N*1*, K, Fn, Fl, Fs, Pl, Ps, N*3*, Kp, lr, E*. Finally, we set $Nseed = 1200,\ Lseed = 100,\ N1 = 300,\ K = 3,\ Fn = 4,\ Fl = 3,\ Fs = 1,\ Pl = 3,\ Ps = 1,\ N3 = 500,\ Kp = 0.96.\ lr = 0.001,\ \mu = 0.9.$

3.3. Comparison results

The aim of our model is to accurately predict the ratings of movies before they are released. As mentioned in the dataset part, we design our experiments by choosing the movies released between 2003 and 2013 as training samples and leave the movies released after 2013 as testing samples.

Table 1
Holdout Predictive Performances on 252 Movies Released After 2013 under different percentage of training dataset; MSE = mean square error; MLP = Multiple Layer Perception; 1DCNN = one dimensional convolutional neural network; DBN = Deep belief network; 1DCNN-SVR = One dimensional convolutional nueral network based support vector regression; LSTM = Long short term memory neural network; LSTM-LR = Long short term memory neural network based linear regression.

Experimental Methods	MSE(50%)	MSE(75%)	MSE(100%)
Multiple Regression	0.835	0.833	0.825
KNN Regression	0.862	0.838	0.835
Hypergrah Regression [7]	0.832	0.804	0.776
Kernel-1 [3]	0.88	0.86	0.848
Support Vector Regression	1.032	0.997	0.93
Decision Tree Regression	1.03	0.9	0.88
Random Forest Regression	0.85	0.79	0.74
Adaboosting	0.91	0.874	0.867
Xgboosting	0.849	0.758	0.737
Gradientboosting	0.774	0.744	0.728
DBN	0.801	0.788	0.765
MLP	0.761	0.736	0.718
LSTM	0.77	0.755	0.721
LSTM-LR	0.766	0.744	0.701
1DCNN	0.743	0.722	0.693
1DCNN-SVR	0.755	0.741	0.722
Ours	**0.71**	**0.676**	**0.628**

We apply different percentages (50%,75%,100%) of training data to train all the models and predict the rating values of testing samples respectively. We evaluate predictive performance of different approaches by the mean squared error(MSE) on the movie ratings and repeat each experiment three times to calculate the average MSE. The results are shown in Table 2.

From the Table 1, we can find that our model consistently outperforms all the comparison methods under different percentages of training samples. Our model achieves 0.71,0.676 and 0.628 mean square error respectively under 50%, 75% and 100% training samples. Compared to 1DCNN which achieves the best performance among all the comparison methods, our model improves predictive performance by 4.4%, 6.3% and 9.3% respectively under 50%, 75% and 100% training samples. The five deep learning methods(MLP, LSTM, LSTM-LR, 1DCNN and 1DCNN-SVR) outperforms other non-deep learning methods while deep belief network does not present better performance. It is notable that the combination of 1DCNN and SVR(1DCNN-SVR) performs poorer than the base 1DCNN model which means the SVR component in neural network is ineffective in our case. Among the non-neural network baselines, gradient boosting achieves the best performance which is a bit worse than the MLP method. The performance of 1DCNN is better than the MLP consistently which confirms the effectiveness of convolutional layer. Besides the traditional regression performance index MSE, we also design another measurement index for comparison. We firstly calculate the absolute value *Abserror* of each predictive error in testing samples. Then we set a threshold θ and count number of the testing samples whose *Abserror* is smaller than θ. We call the number as 'hit number' and calculate the ratio(called 'hit ratio') between 'hit number' and total number of testing samples. It is notable that the 'hit ratio' is an important evaluation metric in our experiment. Although MSE is a convincing metric to evaluate the model performance, sometimes it does not reflect the true result. For example, a certain model can get a relatively small MSE by predicting all the ratings in a moderate error range, in this case we can not regard it as an accurate rating predictor. In this context, we regard 'hit ratio' as a supplementary index for MSE. We show the 'hit number' and 'hit ratio' of all the models under four different θ values(0.3, 0.5, 0.8, 1.0). The final results are shown in Table 2. From this table we can find that the overall performance of all methods is similar to MSE measurement while our

Table 2
Holdout Predictive Performances on 252 Movies Released After 2013 under different percentage of training dataset; MSE = mean square error; MLP = Multiple Layer Perception; 1DCNN = one dimensional convolutional neural network; ; DBN = Deep belief network; 1DCNN-SVR = One dimensional convolutional nueral network based support vector regression; LSTM = Long short term memory neural network; LSTM-LR = Long short term memory neural network based linear regression.

Experimental methods	$\theta(0.3)$	$\theta(0.5)$	$\theta(0.8)$	$\theta(1.0)$
Multiple Regression	54(0.19)	90(0.357)	132(0.825)	175(0.694)
KNN Regression	70(0.27)	94(0.373)	138(0.547)	167(0.66)
Hypergrah Regression [7]	62(0.246)	103(0.408)	146(0.579)	184(0.73)
Kernel-1 [3]	73(0.289)	84(0.33)	143(0.567)	170(0.674)
Support Vector Regression	50(0.198)	88(0.349)	118(0.468)	175(0.694)
Decision Tree Regression	64(0.253)	92(0.365)	139(0.551)	163(0.64)
Random Forest Regression	69(0.273)	102(0.404)	150(0.595)	180(0.714)
Adaboosting	57(0.22)	98(0.388)	136(0.539)	175(0.694)
Xgboosting	63(0.25)	102(0.404)	154(0.611)	183(0.726)
Gradientboosting	71(0.281)	107(0.424)	161(0.638)	184(0.730)
DBN	68(0.269)	101(0.40)	139(0.551)	166(0.658)
MLP	78(0.309)	113(0.448)	158(0.626)	185(0.731)
LSTM	80(0.317)	108(0.428)	152(0.603)	176(0.698)
LSTM-LR	83(0.329)	112(0.444)	155(0.615)	188(0.746)
1DCNN	89(0.353)	108(0.428)	170(0.674)	196(0.777)
1DCNN-SVR	75(0.297)	110(0.436)	165(0.654)	174(0.690)
Ours	**109(0.432)**	**131(0.519)**	**184(0.73)**	**221(0.853)**

model still achieves the best performance consistently over all the θ values. The hit ratio of our model achieves 43.2%, 51.9%, 73% and 85% ($\theta = 0.3, 0.5, 0.8, 1.0$) which improves 7.9%, 9.1%, 6.4% and 7.6% performance compared to the best baseline(1DCNN). It is notable that the hit ratio of our model achieves up to 43.2% even though the θ is set as 0.3 which is a relatively small value. The hit ratio results show that our model can indeed predict the movie ratings at a high precision. From both the two tables, we can find that our model outperforms the base 1DCNN model consistently under two different evaluation metrics which confirms the effectiveness of the artificial samples generators(G_1 and G_2).

Additionally, we also evaluate the trend of predictive performance with varied values of eight hyper-parameters (learning rate *lr*, number of K nearest neighbors in G2 *K*, number of neurons in N3 layer *N*3, numbers of neurons in N1 layer *N*1, number of seed vectors *Nseed*, length of seed vector *Lseed*, values of weight in loss $\mathcal{L}_1$, keep probability *Kp*). The results are shown in Fig.3. From the figure, we can find that the hyper-parameter settings is important to the final performance of our model. Among all the eight chosen hyper-parameters, our model is more sensitive to number of K nearest neighbors in G2, number of neurons in N3 layer and the number of seed vectors than the other hyper-parameters. As for the parameter μ, if it is less than the optimal value 0.8, the performance of our model decreases significantly which means the balance between losses is very important. We can find the MSE can fluctuate a lot if the hyper-parameter setting is set far away from the optimal value.

Furthermore, we also show the training loss and testing mean square error in Fig. 4(a). From this figure, we can find that both the training loss and testing mse descend fast in the first 300 epochs and slower in the following epochs. Finally, the training loss and testing mse nearly converge to the same value which means the overfitting problem is alleviated well by the artificial samples. Although the experimental results in Tables 1 and 2 have already shown the effectiveness of our model, we decide to further confirm effectiveness of artificial generators in our model by conducting ablation study. To conduct ablation study, we firstly compare the predictive performance (testing mse) between our model and discriminator which excludes the artificial generators(G_1 and G_2). The predictive performance of two models via the trend of epochs are shown in Fig. 4(b). From this figure, we can clearly observe that our model outperforms consistently than the discriminator without artificial generators(approximately 0.07 lower in MSE value). We also conduct another ablation study to confirm the effectiveness of feature transformation(auto-encoder). We compare the predictive performance of our model(with auto-encoder for feature transformation) and our model without autoencoder(using raw feature). Similar to the artificial generators, we also show the predictive performance of two models via the trend of epochs are shown in Fig. 4(c). From the figure, we can find that our model can achieve approximately 0.03 lower value in MSE compared to the model excluding autoencoder. Comparing Fig. 4(b) and 4(c), we can find that artificial generators play a more important role than the autoencoder in final predictive performance(decrease 0.07 and 0.03 in MSE value of final predictive performance).

3.4. Conclusion and future

In this paper, we present a generative convolutional neural networks based regression model for movie rating prediction before a movie releases. The approach features using only movies' intrinsic pillars which are all obtainable even before movie production, for final the prediction. Owing to the limited number of available movies (many movies lack plot information and adequate historical features), we propose the generative model that can increase the training samples by generating high-quality artificial samples. Our model consists of three components: artificial feature generator G_1, artificial label generator G_2, and discriminator where G_1 and G_2 are used to generate artificial samples which can alleviate the training samples size problem, the discriminator is a traditional 1DCNN model which can capture the complex relationships between different features.

Experimental results on IMDB dataset demonstrate that our model outperforms a series of baseline methods on both two measurement index MSE(0.628) and Hit ratio(85.3%). Additionally, we also compare our model with discriminator only(partial model which excludes the generator parts) during the whole training process, the experimental results show that our model outperforms the partial model consistently which confirms the effectiveness of our sample generators(G_1, G_2). Furthermore, we also compare our model with partial model excluding autoencoder during the whole training process, the experimental result also show that the partial model performs poorer than our model which confirms the effectiveness of autoencoder.

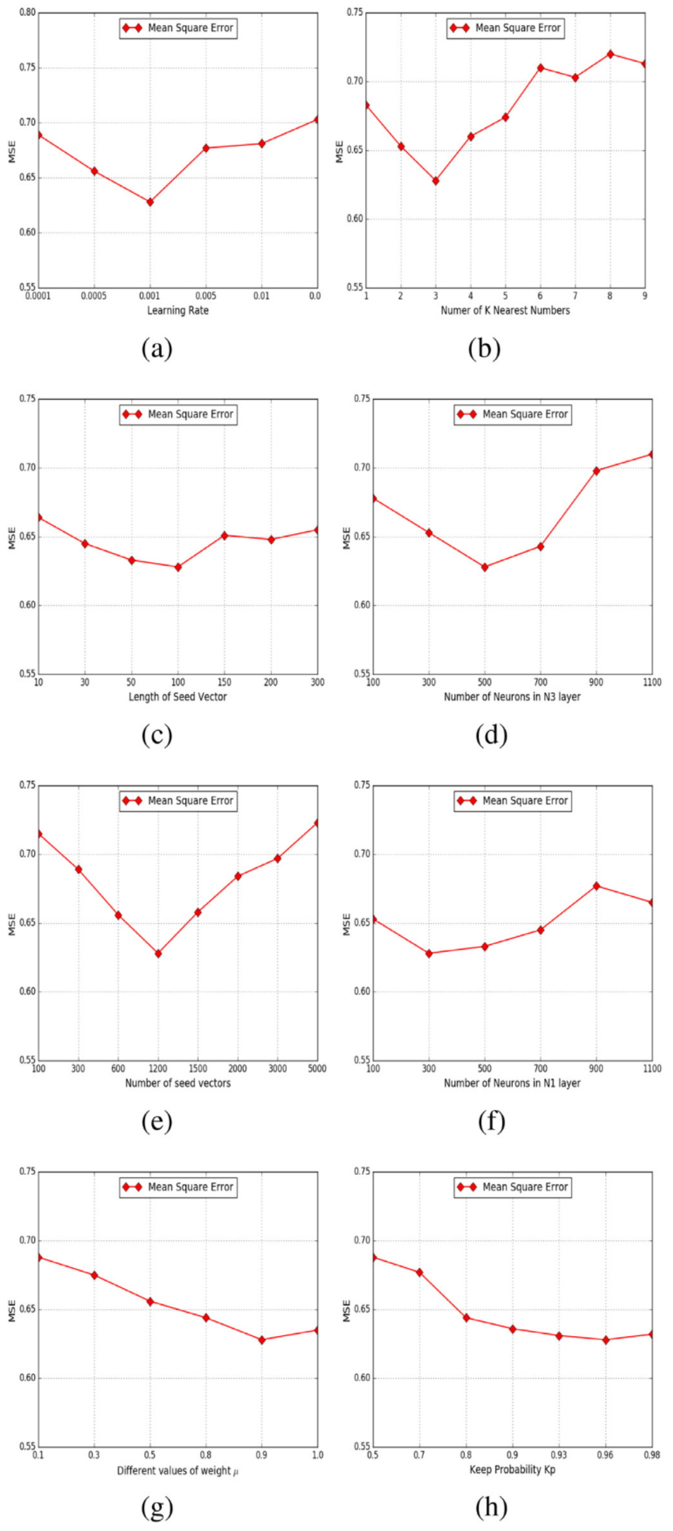

Fig. 3. (a) Trend of Performance Metrics with different learning rates; (b)Trend of performance metrics with different number of K nearest neighbors regression in label generator G_2; (c)Trend of Performance Metrics with different length of seed vector in G_1;(d) Trend of Performance Metrics with different numbers of neurons in N3 layer; (e) Trend of Performance Metrics with different numbers of seed neurons in G_1; (f)Trend of Performance Metrics with different numbers of neurons in N1 layer; (g)Trend of Performance Metrics with different values of weight μ; (h)Trend of Performance Metrics with different values of keep probabilities.

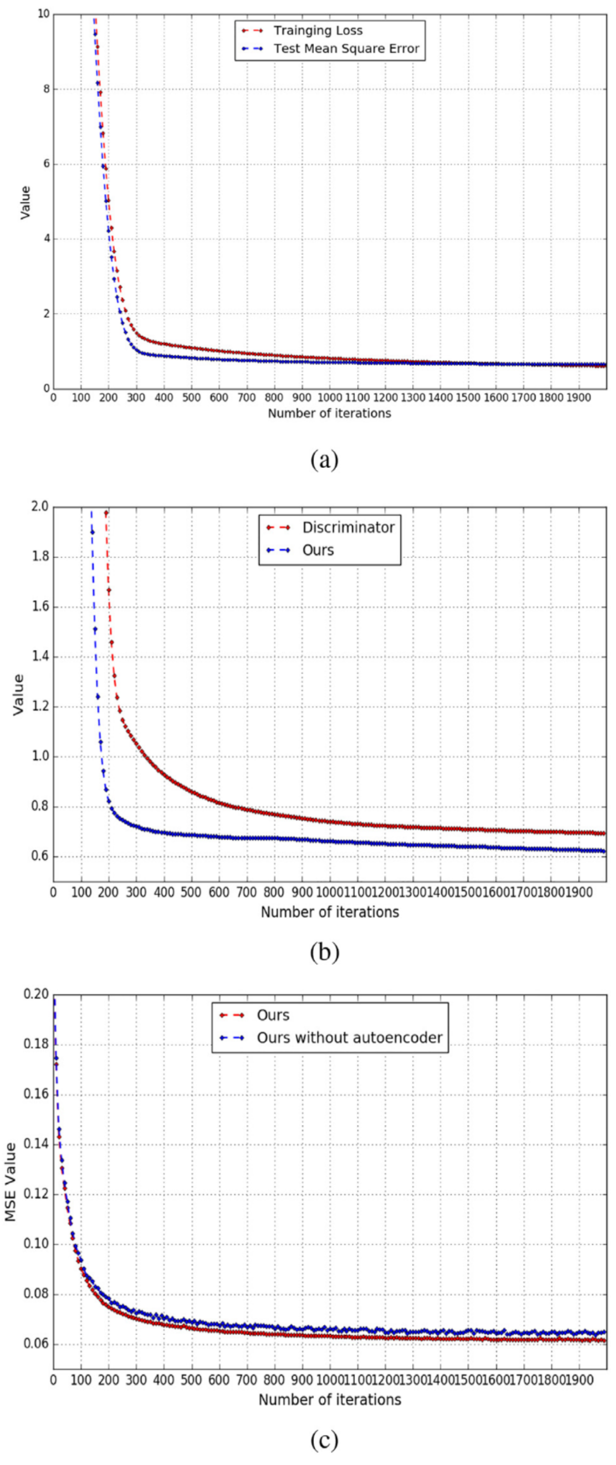

Fig. 4. (a)comparison between training loss testing mean square error during the whole training process; (b)comparison between the mean square error(MSE) of our model and discriminator(excluding the artificial generators G_1 and G_2) during the whole training process;(c)comparison between the mean square error(MSE) of our model and the model excluding autoencoder during the whole training process.

Our current work focuses on exploiting the general attributes and historical information of movies. In future work, we will take the word of mouth ((such as discussion in twitter, facebook and search activity in google)WOM) before the movie releases into consideration to further improve the prediction accuracy. Additionally, distinguishing the movies of which the ratings are improved by the artificial ratings posted from paid posters can be an open research problem in the future. As our model achieves satisfactory performance in this paper, it can also be extended to solve other prediction tasks where the amount of training samples is limited.

References

[1] D. Basak, S. Pal, D.C. Patranabis, Support vector regression, Neural Inf. Process. Lett. Rev. 11 (10) (2007) 203–224.
[2] Q. Diao, M. Qiu, C.-Y. Wu, A.J. Smola, J. Jiang, C. Wang, Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars), in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 193–202.
[3] J. Eliashberg, S.K. Hui, Z.J. Zhang, Assessing box office performance using movie scripts: a kernel-based approach, IEEE Trans. Knowl. Data Eng. 26 (11) (2014) 2639–2648.
[4] O.B. Fikir, I.O. Yaz, T. Özyer, A movie rating prediction algorithm with collaborative filtering, in: Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, IEEE, 2010, pp. 321–325.
[5] P.-Y. Hsu, Y.-H. Shen, X.-A. Xie, Predicting movies user ratings with imdb attributes, in: International Conference on Rough Sets and Knowledge Technology, Springer, 2014, pp. 444–453.
[6] F.J. Huang, Y. LeCun, Large-scale learning with svm and convolutional netw for generic object recognition, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006.
[7] S. Huang, D. Yang, B. Liu, X. Zhang, Regression-based hypergraph learning for image clustering and classification, arXiv:1603.04150v1 (2016).
[8] S. Kabinsingha, S. Chindasorn, C. Chantrapornchai, Movie rating approach and application based on data mining, Int. J. Eng. Innovative Technol. (IJEIT) Vol. 2 (2012).
[9] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980v1 (2014).
[10] A. Kose, C. Kanbak, N. Evirgen, Performance comparison of algorithms for movie rating estimation, arXiv:1711.01647v1 (2017).
[11] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[12] C.-L. Liu, W.-H. Hsaio, C.-H. Lee, G.-C. Lu, E. Jou, Movie rating and review summarization in mobile environment, IEEE Trans. Syst. Man. Cybern.Part C (Appl. Rev.) 42 (3) (2012) 397–407.
[13] R. Navarathna, P. Lucey, P. Carr, E. Carter, S. Sridharan, I. Matthews, Predicting movie ratings from audience behaviors, in: Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on, IEEE, 2014, pp. 1058–1065.
[14] X. Ning, L. Yao, X. Wang, B. Benatallah, Calling for response: automatically distinguishing situation-aware tweets during crises, in: International Conference on Advanced Data Mining and Applications, Springer, 2017, pp. 195–208.
[15] X. Ning, L. Yao, X. Wang, B. Benatallah, S. Zhang, Data-augmented regression with generative convolutional network, in: International Conference on Web Information Systems Engineering, 2018.
[16] A. Oghina, M. Breuss, M. Tsagkias, M. de Rijke, Predicting imdb movie ratings using social media, in: European Conference on Information Retrieval, Springer, 2012, pp. 503–507.
[17] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.
[18] W. Schmit, S. Wubben, Predicting ratings for new movie releases from twitter content, in: 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis Wassa 2015, 2015, p. 122.
[19] D.S. Tan, S. See, T.J. Tiam-Lee, Automatic rating of movies using an arousal curve extracted from video features, in: Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), 2014 International Conference on, IEEE, 2014, pp. 1–6.
[20] C. Trabelsi, G. Pasi, Mrra: a new approach for movie rating recommendation, in: International Conference on Flexible Query Answering Systems, Springer, 2017, pp. 84–95.
[21] D.T. Wijaya, S. Bressan, A random walk on the red carpet: rating movies with user reviews and pagerank, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, ACM, 2008, pp. 951–960.
[22] B. Wu, K. Li, F. Ge, Z. Huang, M. Yang, S.M. Siniscalchi, C.-H. Lee, An end-to-end deep learning approach to simultaneous speech dereverberation and acoustic modeling for robust speech recognition, IEEE J. Sel. Top. Signal Process. 11 (8) (2017) 1289–1300.
[23] S. Zhang, L. Yao, X. Xu, AutoSVD++: An efficient hybrid collaborative filtering model via contractive auto-encoders, arXiv:1704.00551v1 (2017).
[24] G. Zhao, X. Qian, X. Xie, User-service rating prediction by exploring social users' rating behaviors, IEEE Trans. Multimedia 18 (3) (2016) 496–506.
[25] L. Zhao, Z. Lu, S.J. Pan, Q. Yang, Matrix factorization+ for movie recommendation., in: IJCAI, 2016, pp. 3945–3951.
[26] W. Zheng, X. Zhu, G. Wen, Y. Zhu, H. Yu, J. Gan, Unsupervised feature selection by self-paced learning regularization, Pattern Recognit. Lett. (2018), doi:10.1016/j.patrec.2018.06.029.
[27] X. Zhu, S. Zhang, R. Hu, Y. Zhu, et al., Local and global structure preservation for robust unsupervised spectral feature selection, IEEE Trans. Knowl. Data Eng. 30 (3) (2018) 517–529.