Data-Augmented Regression with Generative Convolutional Network

Xiaodong Ning[1], Lina Yao[1], Xianzhi Wang[2], Boualem Benatallah[1], Shuai Zhang[1], and Xiang Zhang[1]

[1] Universit of New South Wales
[2] University of Technology Sydney
xiaodong.ning@student.unsw.edu.au lina.yao@unsw.edu.au

Abstract. Generative adversarial networks (GAN)-based approaches have been extensively investigated whereas GAN-inspired regression (i.e., numeric prediction) has rarely been studied in image and video processing domains. The lack of sufficient labeled data in many real-world cases poses great challenges to regression methods, which generally require sufficient labeled samples for their training. In this regard, we propose a unified framework that combines a robust autoencoder and a generative convolutional neural network (GCNN)-based regression model to address the regression problem. Our model is able to generate high-quality artificial samples via augmenting the size of a small number of training samples for better training effects. Extensive experiments are conducted on two real-world datasets and the results show that our proposed model consistently outperforms a set of advanced techniques under various evaluation metrics.

1 Introduction

Classification and regression are two main types of machine learning applications. While previous research mostly focuses on classification tasks, regression has received less attention. Although a regression task can generally be converted into a multi-classification task by approximating continuous variables using discrete classes, regression can provide more meaningful and accurate insights in many real-world problems such as house price prediction [14], stock price forecasting [13], crime rate inference [11], and movie box prediction [10]. Some commonly used regression methods include parametric and semi-parametric spatial hedonic models, state frequency memory (SFM) recurrent network[3], (kernel-based) regression models[2], and Hodrick–Prescott filter and regression hybrid models.

Different from the above work, we aim to propose a unified framework to solve general regression problems instead of targeting a specific topic. Our framework integrates three parts together: feature preprocessing, feature transformation, and numeric prediction model. Feature preprocessing is responsible for converting the raw meta-data into a unified feature format. The feature transformation part applies a robust autoencoder to eliminate outliers and noise and to extract high quality, non-linear latent features from the original feature. In addition, we alleviate the problem of small training samples by generating artificial samples based on a generative convolutional neural network(GCNN), a new variant of generative adversarial networks (GAN) [9] that we propose specially for regression tasks.

Recently, generative adversarial networks(GAN) has attracted a lot of attention for its capability of generating photorealistic images or videos that help visualize new interior/industrial designs, daily commodities or items for scenes in computer games. The basic GAN is implemented by a system of two neural networks contesting with each other in a zero-sum game framework. Many variants of GAN have been proposed since its invention. For example, Radford et al.[15] propose a deep convolutional generative adversarial network (DCGANs) that regards parts of the generator and discriminator networks as feature extractors in CNN. Liu et al.[6] propose a coupled generative adversarial network (CoGAN) for learning a joint distribution of multi-domain images. While most of the previous investigations about GAN focus on solving the classification or image generation problems, our model aims to utilize the generator for data-augmentation in regression tasks with a small number of training samples.

We make the following contributions in this paper:

- We propose a unified framework that combines feature transformation and numeric prediction for general regression tasks. To the best of our knowledge, our work is the first to address the regression problem using the idea of GAN in a data-augmented manner.
- We design a generative convolutional neural network based regression model to solve the continuous numeric prediction problem with small labeled samples. Our model is able to effectively augment the available number of training data by generating high-quality artificial samples.
- We conduct comprehensive experiments on two real-world datasets and demonstrate that our proposed method consistently outperforms several non-deep learning and deep learning methods.

2 A Unified Framework

We propose a unified framework that consists of three main parts: feature extraction, feature transformation and a generative convolutional neural network. The architecture of our model is shown in Figure 1, where the meta-data is fed into our model first, then the original features are extracted and transformed via robust autoencoder. After that, the artificial samples are produced from the artificial feature generator(G1) and artificial label generator(G2). And finally, both the artificial samples and real samples are fed into discriminator for training. We will describe the details in the following subsections.

2.1 Feature Preprocessing

We use two datasets to evaluate our model: IMDB dataset and American Community Crime dataset. We extract two types of features from meta-data from the datasets: the intrinsic attributes of each movie and the community properties such as population distribution and law enforcement distribution. To reduce the influence of missing values and noises in the extracted features on the final performance of the model, we first process it before feeding it into the prediction model. Our first step is to fill up the missing

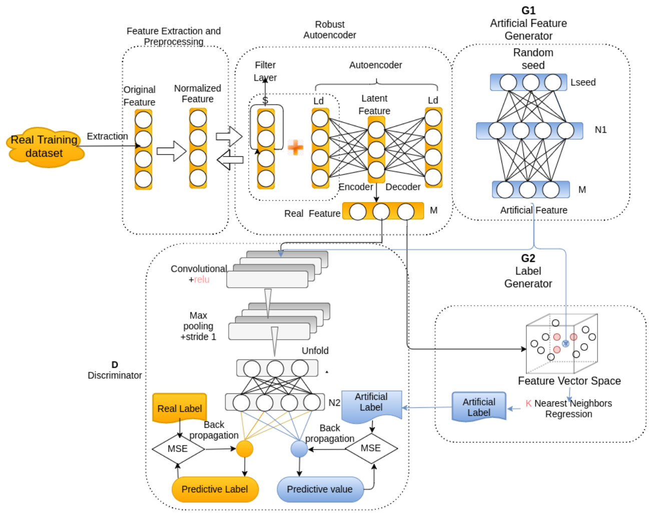


Fig. 1: Overall Framework

feature values by replacing the ‘Null’ values with the median value of all samples. As the value of different features can differ in wide range, we convert them into a unique range $0-1$ by feature normalization. Now, we get a complete feature vector of all the samples where each feature value falls within $0-1$.

2.2 Feature Transformation

We use a robust autoencoder network, which is a variant of the normal autoencoder proposed by Zhou et al [7], to transform features and to discover high quality, non-linear features while eliminating outliers and noise to clean the training data. The robust autoencoder differs from normal autoencoders in adding a filter layer into its network. The filter layer is used to cull out the anomalous parts of the data that are difficult to reconstruct so as to represent the remaining portion of the data by the low-dimensional hidden layer with a small reconstruction error. First, it splits the input data X into two parts $X = L_D + S$, where L_D represents the part of the input data that is well represented by the hidden layer of the auto-encoder and S contains the noise and outliers which are difficult to reconstruct. By removing the noise and outliers from X, the auto-encoder can recover the remaining L_D more accurately. The loss function for a given input X can be either the l_1 norm or $l_{2,1}$ norm of S balanced against the reconstruction error of L_D. In a data-driven manner, we choose the $l_{2,1}$ norm in this paper. The overall process of the robust auto-encoder is shown in equation 1, where E_θ denotes the encoding process of auto-encoder, D_θ denotes the decoding process of auto-encoder, and λ tunes the weight between reconstruction error of L_D and sparsity of S.

$$\min_{\theta} ||L_D - D_\theta(E_\theta(L_D))||_2 + \lambda||S||_{2,1}$$
$$s.t. X - L_D - S = 0 \tag{1}$$

In particular, $l_{2,1}$ norm is defined in equation 2, where $S \in R^{m \times n}$.

$$||S||_{2,1} = \sum_{j=1}^{n} ||S_j||_2 = \sum_{j=1}^{n} (\sum_{i=1}^{m} |S_{ij}|^2)^{1/2} \tag{2}$$

The objective of equation 1 is optimized by L_D and S, which are trained independently through iterations in a similar procedure as that described in [7]. The optimization of L_D is similar to the optimization of a traditional autoencoder while the minimization of the $l_{2,1}$ norm of S is a complicated proximal problem [7]. We present a modified minimization process of $||S||_{2,1}$ in Algorithm 1. This algorithm differs from [7] in that we set the value of $S[i,j]$ as a random value within the range $[-e_j, e_j]$ instead of setting it zero when the e_j is less than λ. By doing so, we can prevent the over-optimizing problem of $||S||_{2,1}$ and its influence on the autoencoder performance. Processed by the robust autoencoder, the original feature is transformed into an M-dimensional feature vector.

Algorithm 1 Random Proximal Method

```
Input: $S \in R^{mxn}, \lambda$
  for j in 1 to n do
      e$_j = (\sum_{i=1}^{m} |S[i,j]|^2)^{1/2}$
    if $e_j > \lambda$ then
      for i in 1 to m do
          $S[i,j] = S[i,j] - \lambda \frac{S[i,j]}{e_j}$
      end for
    else
      for i in 1 to m do
          $S[i,j] = Random[-e_j, e_j]$
      end for
    end if
end for
Output: S
```

2.3 Generative Convolutional Neural Network based Regression Model

In recent years, deep learning has been proven to achieve promising performance in various domains such as twitter classification [5], rating prediction[1] and brain disease diagnosis [4], etc. However, the requirement for a large amount of training data prevents deep learning networks from being applied into many real problems. In this regard, we propose a generative convolutional neural networks based regression model inspired by the generative adversarial network (GAN)[9]. Instead of contesting between generator and discriminator in GAN, our model utilizes a generator to generate artificial samples and combine them with real samples to co-train the entire network to solve the limited training samples problem. As shown in figure 1, our regression model consists of three components: artificial feature generator(G1), artificial label generator(G2) and discriminator(D).

Artificial Feature Generator G1 The artificial feature generator G1 works as follows. Firstly, we initialize $Nseed$ seed vectors with $Lseed$ length where each vector point is generated randomly from the range of (0, 1). Then, the seed vectors are fed into the two fully-connection layers (the neurons in each layer are $N1$ and M). We apply Leaky Relu activation function on the dot product results generated from each layer.

Via the two fully connected layers, the random seed vector will be transformed into an M-dimensional artificial feature vector (with the same length as the real transformed feature vector from the robust auto-encoder).

Artificial Label Generator G2 We design an artificial label generator G2 to assign labels for the artificial features generated from G1. As the artificial feature vector and real transformed feature vector share the same dimension, we apply the weighted K nearest neighbors regression to label the artificial samples using the real training samples. The distance between artificial samples and real samples are calculated as their Euclidean distance, as described in equation 3, where D_{ij} denotes the distance between i_{th} artificial sample and j_{th} real sample, M denotes the length of feature, F_{ik} denotes k_{th} feature value of i_{th} artificial sample, and F_{jk} denotes k_{th} feature value of j_{th} real sample.

$$D_{ij} = \sum_{k=1}^{M} \frac{(F_{ik} - F_{jk})^2}{M} \tag{3}$$

Then, we apply weighted knn regression to calculate the labels for artificial samples as follows:

$$A_{ti} = \frac{\sum_{j=1}^{K} D_{ij} * L_j}{\sum_{j=1}^{K} D_{ij}} \tag{4}$$

where A_{ti} denotes the generated label of the i_{th} artificial feature; K denotes the top K nearest neighbors to the i_{th} artificial feature; D_{ij} denotes the distance between i_{th} artificial sample and j_{th} nearest real sample; L_j denotes the label value of j_{th} nearest real sample

Discriminator The discriminator is a one-dimensional convolutional neural network (1DCNN) based regression network where the artificial samples feed in. It consists of four layers: one convolutional layer, one 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. Along the way, the dot product is taken between the filters and chunks of the input features. Filters are used to generate the feature vectors in each filter length. The same padding is chosen for the convolutional layer in order to keep the same size of output as input. In this way, the feature vector is projected into a stack of feature maps (vector maps in our work). Followed by the convolutional layer, we add one max-pooling layer. The convolved and max-pooled feature vectors will be unfolded and fed into two fully-connected layers (the neurons in each layer are $N2$ and 1) applied for the high-level reasoning. 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 label. 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 $\mathcal{L}_1$ and $\mathcal{L}_2$, where $\mathcal{L}_1$ represents the mean square rrror of artificial generated samples, and $\mathcal{L}_2$ represents the mean square error of real samples. Additionally, a multiplying factor μ(0-1) is used in $\mathcal{L}_1$ to tune the weight between two loss functions.

Network Training In our model, only feature generator (G1) and discriminator should be trained. The feature generator G1 and discriminator are trained via back-propagation based on the Adam optimizer. 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. In comparison, the feature generator G1 is trained only with artificial samples($\mathcal{L}_1$). The G1 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 G1 will also improve the training effect of the discriminator.

By optimizing the two loss functions, feature generator G1 learns how to generate high-quality artificial features similar to the real samples, and the discriminator learns to predict the final labels accurately.

3 Experiment

3.1 Dataset

Dataset. We choose two datasets to evaluate our model. One is the IMDB dataset[3], where movie attributes and plot information are obtained from the IMDB dataset in Kaggle and the IMDB website, respectively. We keep among all movies $1,471$ US movies released after 2003 to ensure each has some plot descriptions and the history ratings and box office of directors and actors. Each movie record contains attributes including facebook likes of directors and cast, genres, country, MPAA rating[4], release date, budgets of the movie, box office records of directors and actors and plot latent topics. 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. The movies are then divided into two part: training samples(1219 movies released between 2003-2013) and testing samples(252 movies released after 2013).

The second dataset is the American community crime rate dataset[5]. This dataset contains various attributes of each community such as the percent of the population considered urban, the median family income, per capita number of police officers, etc. These attributes are used to predict the per capita violent crime for each community. Among the total 1994 samples, we use 1000 as training samples and keep the remaining 994 as testing samples.

[3] https://www.kaggle.com/tmdb/tmdb-movie-metadata

[4] https://en.wikipedia.org/wiki/Motion_Picture_Association _of_America_film_rating_system#MPAA_film_ratings

[5] https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

Table 1: Mean absolute error of each approach on IMDB and Community Crime datasets under different percentage of training samples; MAE = Mean Absolute Error

Datasets	IMDB			Community Crime		
Experimental Methods	MAE(50%)	MAE(75%)	MAE(100%)	MAE(50%)	MAE(75%)	MAE(100%)
RF Regressor	0.914	0.88	0.87	0.104	0.099	0.095
Gradientboosting	0.89	0.872	0.87	0.109	0.103	0.10
Adaboosting	0.921	0.91	0.9	0.1369	0.133	0.121
Xgboosting	0.945	0.937	0.935	0.109	0.105	0.103
SVR	0.882	0.865	0.86	0.126	0.125	0.123
Kernel-1[10]	0.86	0.841	0.83	0.121	0.117	0.111
Hypergrah Regression[12]	0.86	0.83	0.822	0.110	0.101	0.098
Deep Belief Network	0.78	0.77	0.76	0.112	0.105	0.103
MLP	0.801	0.765	0.751	0.101	0.097	0.095
1DCNN	0.812	0.763	0.744	0.096	0.095	0.091
1DCNN-SVR[2]	0.788	0.754	0.738	0.101	0.096	0.093
Ours	**0.712**	**0.701**	**0.68**	**0.093**	**0.090**	**0.089**

3.2 Comparison Methods

We set the default settings of our model as following:$M = 55$, $Nseed = 500$, $Lseed = 200$, $K = 8$, $\mu = 1.0$ for IMDB;$M = 90$, $Nseed = 200$, $Lseed = 60$, $K = 5$, $\mu = 0.9$ for Crime. We also tune the optimal parameters of each comparison method respectively for the two datasets for a fair comparison.

- *Random Forest Regressor (RF)* with 200 estimators(IMDB); 20 estimators(Crime).
- *Gradientboosting (Gra)* with 30 estimators(IMDB); 25 estimators(Crime).
- *Adaboosting (Ada)* with base estimator as the decision tree regressor, 50 estimators(IMDB); 50 estimators(Crime)
- *Xgboosting (Xg)*with 100 estimators(IMDB); 30 estimators(Crime).
- *Support Vector Regressor(SVR)* with penalty parameter as 0.8 (IMDB); penalty parameter as 1.0 (Crime).
- *Kernel-1 Regression method* is a kernel-based approach with an improved version of KNN regression [10]. In our paper, we utilize the recommended parameter settings in the paper for both two datasets.
- *Hypergraph Regression* is a regression version of the hypergraph classification [12]. We define a hyperedge by each sample and its K nearest neighbors to form a hypergraph. The weight of each hyperedge is calculated using the mean similarities of pairs in this hyperedge. The predicted label of i_{th} sample is calculated using the weighted average ratings of the samples belonging to the same hyperedge with i_{th} sample. We set K as 10 and 8 for IMDB and Crime;
- *Multiple Layer Perceptron (MLP)* with three layers (100,300,1 neurons in each layer) (IMDB); with three layers(200,350,1 neurons in each layer) (Crime).
- *Deep Belief Network Regression* A deep belief regression network is proposed 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

Table 2: Hit ratio of each approach on IMDB and Community Crime datasets under different threshold $\theta(3\%, 5\%, 8\%, 10\%, 15\%)$;

Datasets	IMDB					Community Crime				
Experimental Methods	3%	5%	8%	10%	15%	3%	5%	8%	10%	15%
RF Regressor	0.18	0.27	0.416	0.535	0.738	0.034	0.062	0.095	0.129	0.181
Gradientboosting	0.19	0.29	0.452	0.523	0.742	0.04	0.063	0.090	0.115	0.183
Adaboosting	0.11	0.23	0.388	0.46	0.66	0.031	0.043	0.071	0.089	0.135
Xgboosting	0.16	0.277	0.42	0.53	0.789	0.026	0.054	0.088	0.111	0.176
SVR	0.18	0.29	0.468	0.567	0.785	0.033	0.056	0.075	0.09	0.141
Kernel-1[10]	0.16	0.26	0.44	0.531	0.747	0.025	0.041	0.070	0.078	0.138
Hypergrah Regression[12]	0.179	0.288	0.463	0.555	0.779	0.032	0.049	0.084	0.092	0.151
Deep Belief Network	0.177	0.30	0.471	0.588	0.788	0.031	0.057	0.098	0.130	0.166
MLP	0.183	0.285	0.489	0.544	0.781	0.044	0.058	0.101	0.141	0.183
1DCNN	0.202	0.293	0.468	0.561	0.787	0.050	0.07	0.111	0.148	0.20
1DCNN-SVR	0.202	0.30	0.471	0.565	0.791	0.048	0.065	0.107	0.144	0.194
Ours	**0.22**	**0.332**	**0.508**	**0.611**	**0.821**	**0.068**	**0.084**	**0.141**	**0.177**	**0.235**

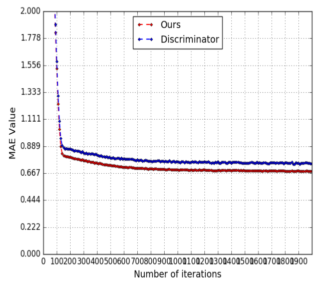
hidden layers (110,200,330 neurons in each layer)(IMDB); with three hidden layers(110,300,200 neurons in each layer)(Crime)

- *One Dimensional Convolutional Neural Network (1DCNN)* with one convolutional layer(5 filters), one max pooling layer and two fully connected layer(300 and 1) for IMDB; one convolutional layer(4 filters), one max pooling layer and two fully connected layer(250,1) for Crime.
- *One Dimensional Convolutional Neural Network with Support Vector Regressor (1DCNN-SVR)* is the regression version of 1DCNN 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.

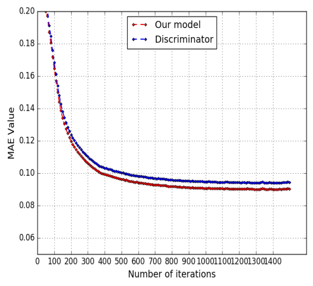
3.3 Comparison Results

Since the aim of our model is to accurately predict labels of new samples in the regression problem, we apply different percentages (50%,75%,100%) of training data to train all the models, predict the rating values of testing samples for both the two datasets and evaluate the performance of different approaches by the mean absolute error (MAE). The holdout results (shown in Table 1) show our model achieves the best performance for both the two datasets with MAE(0.712,0.701,0.68) and MAE(0.093,0.090,0.089), under 50%, 75% and 100% training samples. 1DCNN-SVR performs the best among the all the compared methods in the IMDB task while the 1DCNN achieves the best performance in Crime task. Compared to the other models, our model achieves a 9.6%, 7.02%, 7.85% improvement in the IMDB task and 6.9%, 6.25% and 4.3% in the Crime task, respectively.

Besides the mean absolute error (MAE), we define a new measure for the comparison. We first calculate the absolute percentage error APE of each testing sample. Then we set a threshold θ and count number of the testing samples with a smaller APE than θ. We call it 'hit number' and calculate the ratio (called 'hit ratio') between 'hit number'

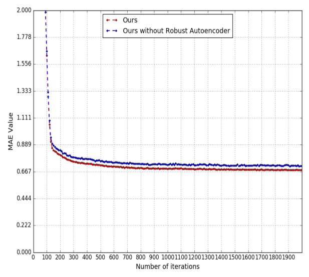


(a)

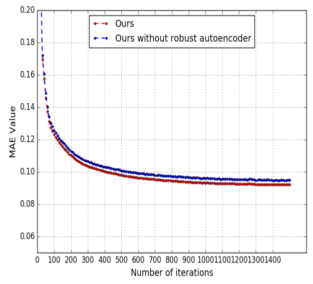


(b)

Fig. 2: Mean Absolute Value(MAE) Comparison between Our Model and Discriminator(excluding the artificial generator G1,G2) in training process for IMDB(a) and Crime(b)



(a)



(b)

Fig. 3: Mean Absolute Value(MAE) Comparison between Our Original Model and Our Model(excluding the robust autoencoder) in training process for IMDB(a) and Crime(b)

and the total number of testing samples. We show the 'hit ratio' of all the approaches under five different θ values(3%, 5%, 8%, 10%, 15%) for both two datasets. The results (shown in Table 2) show our model achieves an average improvement of 3.26% and 2.94% in IMDB and Crime, respectively.

4 Ablation Study

In addition, we carry out the albation study to examine the effectiveness of the components in our model. We conduct comparative experiments by excluding artificial generator and robust autoencoder from our model on two tasks during training process. The MAE of training processes is shown in Figure 2 and Figure 3, respectively. We can observe that the two components indeed improve the training process by influencing the final MAE performance, while artificial generator has more impact on the performance(approximately 7.6% and 3.4% improvement in IMDB and Crime respectively) than robust autoencoder(approximately 4.3% and 3.1% improvement in IMDB and Crime respectively).

5 Conclusions

In this paper, we propose an integrated model for general regression tasks. Our model leverages a robust auto-encoder in combination with a generative convolutional neural network for feature transformation and numeric prediction. Owing to artificial sample generator (G1, G2), our model is capable of handling the numeric prediction tasks with a small size of training samples, which are originally insufficient for traditional deep learning methods. Extensive experiments on two real-world datasets have shown the superior performances of our model over a series of existing advanced techniques.

References

1. NING, XIAODONG, et al. ”Rating Prediction via Generative Convolutional Neural Networks based Regression.” Pattern Recognition Letters (2018).
2. Christmann, Andreas, and Ingo Steinwart. ”Consistency and robustness of kernel-based regression in convex risk minimization.” Bernoulli 13.3 2007.
3. Hu, Hao, and Guo-Jun Qi. ”State-Frequency Memory Recurrent Neural Networks.” ICML. 2017.
4. Suk, Heung-Il, Seong-Whan Lee, and Dinggang Shen. ”Deep ensemble learning of sparse regression models for brain disease diagnosis.” Medical image analysis 2017.
5. Ning, Xiaodong, et al. ”Calling for Response: Automatically Distinguishing Situation-Aware Tweets During Crises.” ADMA, 2017.
6. Liu, Ming-Yu, and Oncel Tuzel. ”Coupled generative adversarial networks.” Advances in neural information processing systems. 2016.
7. Zhou, Chong, and Randy C. Paffenroth. ”Anomaly detection with robust deep autoencoders.” Proceedings of the 23rd ACM SIGKDD, 2017.
8. Gauthier, Jon. ”Conditional generative adversarial nets for convolutional face generation.” Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, 2014.
9. Goodfellow, Ian, et al. ”Generative adversarial nets.” NIPS. 2014.
10. Eliashberg, Jehoshua, Sam K. Hui, and Z. John Zhang. ”Assessing box office performance using movie scripts: A kernel-based approach.” TKDE 26.11 (2014).
11. Gerber, Matthew S. ”Predicting crime using Twitter and kernel density estimation.” Decision Support Systems 61 (2014).
12. Huang, Sheng, et al. ”Regression-based Hypergraph Learning for Image Clustering and Classification.” arXiv preprint arXiv:1603.04150 (2016).
13. Zhang, Liheng, Charu Aggarwal, and Guo-Jun Qi. ”Stock Price Prediction via Discovering Multi-Frequency Trading Patterns.” Proceedings of the 23rd ACM SIGKDD, 2017.
14. Montero, José-María, Román Mínguez, and Gema Fernández-Avilés. ”Housing price prediction: parametric versus semi-parametric spatial hedonic models.” Journal of Geographical Systems 20.1 (2018).
15. Radford, Alec, Luke Metz, and Soumith Chintala. ”Unsupervised representation learning with deep convolutional generative adversarial networks.” arXiv preprint arXiv:1511.06434 (2015).