Hindawi
Shock and Vibration
Volume 2020, Article ID 8893381, 7 pages
https://doi.org/10.1155/2020/8893381

Research Article

Estimating Network Flowing over Edges by Recursive Network Embedding

Liqun Yu,[1] **Hongqi Wang**,[1] **and Haoran Mo**[2]

[1]*School of Economics and Management, Harbin University of Science and Technology, Harbin 150080, China*
[2]*Innopolis University, Innopolis, Russia*

Correspondence should be addressed to Liqun Yu; yuliqun@hrbust.edu.cn

Received 29 September 2020; Revised 21 October 2020; Accepted 2 November 2020; Published 26 November 2020

Academic Editor: Chaoqun Duan

In this paper, we propose a novel semisupervised learning framework to learn the flows of edges over a graph. Given the flow values of the labeled edges, the task of this paper is to learn the unknown flow values of the remaining unlabeled edges. To this end, we introduce a value amount hold by each node and impose that the amount of values flowing from the conjunctive edges of each node to be consistent with the node's own value. We propose to embed the nodes to a continuous vector space so that the embedding vector of each node can be reconstructed from its neighbors by a recursive neural network model, linear normalized long short-term memory. Moreover, we argue that the value of each node is also embedded in the embedding vectors of its neighbors, thus propose to approximate the node value from the output of the neighborhood recursive network. We build a unified learning framework by formulating a minimization problem. To construct the learning problem, we build three subproblems of minimization: (1) the embedding error of each node from the recursive network, (2) the loss of the construction for the amount of value of each node, and (3) the difference between the value amount of each node and the estimated value from the edge flows. We develop an iterative algorithm to learn the node embeddings, edge flows, and node values jointly. We perform experiments based on the datasets of some network data, including the transportation network and innovation. The experimental results indicate that our algorithm is more effective than the state-of-the-arts.

1. Introduction

1.1. Background. Learning the flow direction and amount of the edges for a network has been a critical problem in the network analysis. The edges connect two nodes, and the flow is defined from one node to another connected by the corresponding edge [1–3]. This problem is called the edge flow estimation. In this problem, we already know the network structure, including the sets of the nodes, and the edges between the nodes. We also know the directions and amounts of the flows of some edges, but for the rest edges, the flows are still unknown. The target is to predict the flow direction and amounts of these edges. Some examples of the applications of the edge estimation are given as follows:

(i) For example, in the transportation network, each intersection is a node, and each road is an edge, while the flows along the roads need to be estimated for the purpose of traffic control. According to the historical data of the traffic flows, we know the flows of some roads; however, other read flows need to be estimated. This use case raises the problem of learning edge flows from both the traffic network and existing flows of roads of the network [4, 5].

(ii) Another example is the innovation network analysis, where each research article is a node, and each citation is an edge between two articles. For the task of innovation, it is important to know the flowing of knowledge from an article to other articles. Please note that one article cites another one that does not necessarily mean there is an amount of "knowledge" flowing from the cited article to the citing article. By reading the content of the article which cites the others, we can annotate the "knowledge flowing" if one work is inspired or a following work of the other

research works. However, such annotation by reading is time-consuming and subject to individual annotators, thus it is very necessary to develop an automatic system to estimate the knowledge flow from the article citation network [6–14].

Although the edge flow prediction/estimation is such a critical problem, the works on this direction are very few. Most recently, Ransom et al. [1] designed a new algorithm to predict the edge flow by balancing the amount of flows moving into and out of the nodes. Another condition of this algorithm is that the predicted flow of edges should be consistent to the known flows, for these edges whose flows are already known. In this paper, we proposed a novel edge flow estimation algorithm based on both semisupervised learning and network embedding.

1.2. Our Contributions. In this paper, we build a novel method for the problem of edge flow prediction. This method is both for network embedding and edge flow prediction. The two problems are solved at the same time. We designed a new framework for the learning problem. In this framework, for the first time, we bridge the node embedding and edge flow estimation, by introducing a node value for each node. On the one side, the node value is used to balance the flowing-in and flowing-out amount of a node. It plays the role of measuring the balance of the amount of the node at any moment of the flowing process, i.e., with the incoming flow and outgoing flow changing, the amount of the value in this node remains as the node value. One the other side, the node is employed to regularize the learning of the embedding vectors, thus we impose that the node value can be estimated from the embedding vector by a linear function.

We propose a novel algorithm to learn the embedding vectors and edge flows simultaneously. We model the learning problem as a minimization problem where the embedding vectors reconstruction error of the LSTM embedding model, the node value estimation error from the embedding vectors, and the node value flow amount error are minimized. In the iterative algorithm, the node embeddings, node values, and the edge flow amounts are optimized alternately.

We evaluated the proposed method over benchmark datasets of networks, conducted experiments to reveal the properties of the proposed algorithms, and show its advantage over state-of-the-art methods.

Remark: the superiority of the proposed semisupervised edge flow learning method compared with the traditional semisupervised learning method is listed as follows:

(1) The traditional semisupervised learning methods can only predict the labels of the nodes but are not able to predict the flows of the edges. For the applications discussed in this paper, the traditional semisupervised learning methods are not suitable. Our method is especially designed for these applications.

(2) Traditional semisupervised learning methods can only use the edge information of a graph but ignore the flow information, which is critical for both the node and flow label prediction. However, our method can effectively use them to learn better node embeddings and flow amounts.

1.3. Paper Origination. This paper is organized as follows. In Section 2, we introduce the joint learning framework, with an objective and optimization solution. In Section 3, we experimentally evaluate the performance of the proposed method and conduct studies over its properties. In Section 4, we conclude this paper with some future works.

2. Proposed Method

In this section, we introduce the proposed joint network embedding and edge flow learning framework. In this framework, we propose a deep learning-based network embedding method [15–23] and further use the embedding vectors to estimate the flow amount regarding each node [15]. The flow amount is used to regularize the edge flow learning process.

2.1. Problem Definition. Suppose we have input network, denoted as $G = \{V, E\}$, where $V = \{1, \ldots, n\}$ is the set of n nodes, and $\varepsilon_k = (i, j) \in E$ is an edge linking the i-th and j-th nodes. Here, we assume that $i < j$. For a group of edges, we already know their flows. This set of edges is denoted as $\varepsilon_k \in E_L$. For such an edge ε_k, we define its flow as $f_k \in R$. The direction of the flow is the sign of f_k, and the amount of the flow is the absolute value of f_k. For the other edges, the flows are unknown, and we want to predict them. For these flows, we define a set as E_U. We define a vector as the flows of all the nodes, as $f = [f_1, \ldots, f_{|E|}]^T \in R^{|E|}$. Thus, the prediction of the flows of the nodes is transformed to the solving of f.

2.2. Problem Modeling. We firstly embed each node $i \in V$ to a vector. The dimension of the vector is denoted as d. Moreover, for each node, we define a node value, ϕ_i. We also propose to calculate the node value from the node's embedding vector. After ϕ_i is given, we use it to regularize the estimation of the flow of edges connected to the node. The flow chart of the proposed semisupervised edge flow learning method is given in Figure 1.

2.2.1. Recursive Node Embedding. The network embedding converts the nodes to a set of vectors, denoted as $x_1, \ldots, x_n$, where $x_i \in R^d$. To this end, we want to reconstruct the embedding vector of one node from its linked nodes. The linked nodes are presented as a sequence of nodes. Accordingly, the sequence of neighbouring nodes of the i-th node are given as

$$N_i = \{j | (i, j) \in E \text{ or } (j, i) \in E\}. \quad (1)$$

We firstly sort the nodes in the neighbouring set, N_i, and the sorted set's embedding vector set is

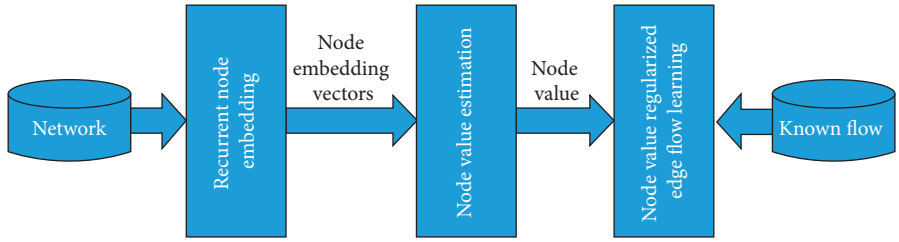

Figure 1: Flow chart of proposed semisupervised flow learning method.

$$S_i = \{x_{i,l}, \ldots, x_{i|Ni|}\}. \quad (2)$$

We apply aln-LSTM to this sequence of embedding vectors. In this model, for an input node's embedding vector, $x_t \in S_i$, its inputs are x_t itself and the output of its previous h_{t-1},

$$h_t = g(x_t, h_{t-1}; \theta), \quad t = 1, \ldots, |\mathrm{Ni}|, \quad (3)$$

where g is the cell function of ln-LSTM and θ is its parameter.

Then, we try to use the output of the cell function from the last node to approximate the embedding vector of the i-th node itself,

$$\mathrm{LSTM}(x_{il}, \ldots, x_{i|Ni|}; \theta) = h_{|Ni|}. \quad (4)$$

Thus, the minimization of the error of the approximation is modelled as

$$\min_{x|_{i=1}^n, \theta} \sum_{i=1}^{n} \left\| x_i - \mathrm{LSTM}(x_{i1}, \ldots, x_{i|Ni|}; \theta) \right\|_F^2. \quad (5)$$

By solving this problem, the learning of the embedding vector and the parameter of the cell function are solved together. With the good quality embedding and cell function parameter, we should be able to approximate the embedding vectors for the neighbouring nodes's embedding vectors.

2.2.2. Node Value Estimation from Recursive Embedding. In out learning framework, the embedding of each node approximated by the recursive model plays two roles, representing the node's neighbor structure and estimating the amount of value hold by the nodes. We define a node value for the i-th node, $\phi_i \in R$. The function of this value is to balance the incoming and outgoing flow of the node. Moreover, it can somehow measure the nature of the node. For example, a node in traffic network at the working time may have more incoming flow, because it is an office area. This value can indicate this node's nature of being in office area. Moreover, we want to use the embedding vector of the node to calculate the value as follows:

$$\phi_i \longleftarrow \varphi(\mathrm{LSTM}(x_{il}, \ldots, x_{i|Ni|}; \theta)), \quad (6)$$

where $\varphi(x) = \sigma(w^T h)$ is the function of node value approximation. This function is actually a single-layer neural network. We proposed to minimize the square error of the approximation over all the nodes,

$$min_{x|_{i=1}^n, \theta, w} \sum_{i=1}^{n} \left\| \phi_i - \varphi(LSTM(x_{i1}, \cdots, x_{i|Ni|}; \theta)) \right\|_F^2. \quad (7)$$

The minimization is performed regarding the node value amount, embedding vectors, recursive model parameters, and the single-layer neural network parameter. In this way, we bridge the learning of the flow amount and embedding of nodes by using an LSTM recursive model.

2.2.3. Flow Prediction by Using Node Value. Given an edge $\varepsilon_k \in E$, we want to predict its flow f_k. To this end, we use the node value as reference. Our assumption is that for any node, the value of the node is controlling the incoming and outgoing flow. To be more specific, for a node, the difference between its incoming flow and outgoing flow should be equal to the node value itself. For this purpose, we define two sets of edges for a node $R_i^+ = \{\varepsilon_k \mid \varepsilon_k \in E, \varepsilon_k = (u, i)\}$ and $R_i^- = \{\varepsilon_k \mid \varepsilon_k \in E, \varepsilon_k = (i, v)\}$. The amount of the incoming flow is $\sum_{k \in R_i^+} f_k - \sum_{k \in R_i^-} f_k$. Since we hope this amount is equal to the node value,

$$\phi_i = \sum_{k \in R_i^+} f_k - \sum_{k \in R_i^-} f_k = \sum_{k=1}^{|E|} \tau_{ik} f_k,$$

where

$$\tau_{ik} = \begin{cases} +1, & \text{if } k \in R_i^+, \\ -1, & \text{if } k \in R_i^-, \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

We also denote the node value of all nodes in a vector $\phi = [\phi_1, \ldots, \phi_n]^T \in R^n$. A matrix is also proposed as $\Phi = [\tau_{ik}] \in \{+1, -1, 0\}^{n \times |E|}$. It is the node-flow mapping matrix. So, equation (8) is transformed to

$$\phi = \Phi f. \quad (9)$$

We minimize the squared approximation error regarding both the edge flow vector and the node value vector,

$$\min_{f, \phi} \|\phi - \Phi f\|_F^2. \quad (10)$$

The final objective and minimization problem is the combination of (5), (7), and (10) as follows:

$$\begin{aligned}
&\min_{f,\phi,x_i|_{i=1}^n,\theta,w}\left\{\sum_{i=1}^{n}\left\|x_i-\mathrm{LSTM}\left(x_{i1},\cdots,x_{i|Ni|};\theta\right)\right\|_F^2\right\}\\
&+\lambda_1\sum_{i=1}^{n}\left\|\phi_i-\varphi\left(\mathrm{LSTM}\left(x_{i1},\cdots,x_{i|Ni|};\theta\right)\right)\right\|_F^2+\lambda_2\|\phi-\Phi f\|_F^2\\
&+\lambda_3\left(\|f\|_F^2+\|\phi\|_F^2+\sum_{i=1}^{n}\|x_i\|_F^2+\|\theta\|_F^2+\|w\|_F^2\right)\Bigg\}\\
&s.t.\quad f_k=\overline{f}_k,\forall k:\ \varepsilon_k\in E_L.
\end{aligned}\tag{11}$$

Here, we also add the l_2 norm regularization terms to the parameters to prevent the over-fitting problem.

2.3. Problem Solution. The solving of the minimization problem is conducted in an iterative algorithm. In each iteration, every parameter is updated sequentially. The others are not updated when one parameter is being updated. In the following subsections, we introduce the optimization of each parameter one by one, while the others are fixed.

2.3.1. Optimization of LSTM Parameters, θ. When the other parameters are fixed, and only the LSTM model parameters are considered, we have the following minimization problem:

$$\min_\theta\left\{o_1(\theta)=\sum_{i=1}^{n}\left\|x_i-\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right\|_F^2+\lambda_1\sum_{i=1}^{n}\left\|\phi_i-\varphi\left(\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right)\right\|_F^2\right\},\tag{12}$$

which can be solved by the ADAM algorithm. To use the ADAM algorithm, we calculate the gradient function of the objective $o_1(\theta)$ regarding θ as follows:

$$\begin{aligned}
\nabla_\theta o_1(\theta)=&-2\sum_{i=1}^{n}\left(x_i-\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right)\times\frac{\partial\,\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)}{\partial\theta}\\
&-2\lambda_1\sum_{i=1}^{n}\left(\phi_i-\varphi\left(\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right)\right)\times\frac{\partial\varphi\left(\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right)}{\partial\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)}\times\frac{\partial\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)}{\partial\theta}+2\lambda_3\theta,
\end{aligned}\tag{13}$$

where $\partial f(x)/\partial x$ is the derivative function of f regarding variable x.

2.3.2. Optimization of Embedding Vectors, $\mathbf{x_i}|_{\mathbf{i=1}}^{\mathbf{n}}$. When the other variables are fixed and only the embedding vectors are considered, we have the following minimization problem:

$$\min_{x_i|_{i=1}^n}\left\{\sum_{i=1}^{n}\left\|x_i-\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right\|_F^2+\lambda_1\sum_{i=1}^{n}\left\|x_i-\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right\|_F^2+\lambda_3\sum_{i=1}^{n}\|x_i\|_F^2\right\}.\tag{14}$$

To solve the vectors, we apply the sequential optimation method and optimize the embedding vectors of the nodes one by one. When one node embedding vector is optimized, others are fixed. When the i-th vector, x_i, is considered, we have the following problem for minimization:

$$\begin{aligned}
&\min_{x_i}\Bigg\{o_2(x_i)=\left\|x_i-\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right\|_F^2\\
&+\sum_{j:\ i\in N_i}\left\|x_j-\mathrm{LSTM}\left(x_{j1},\ldots,x_{j|Ni|};\theta\right)\right\|_F^2+\lambda_1\sum_{j:\ i\in N_i}\left\|\phi_i-\varphi\left(\mathrm{LSTM}\left(x_{i1},\ldots,x_{i|Ni|};\theta\right)\right)\right\|_F^2+\lambda_3\|x_i\|_F^2\Bigg\}.
\end{aligned}\tag{15}$$

Table 1: Dataset summary.

Dataset	# Node	# Edge	Node	Edge	Reference
Minnesota road network	2642	3303	Intersection	Road	[19]
US power grid network of KONECT	4941	6593	Consumer	Transmission line	[24]
Water irrigation network of Balerma, Spain	447	454	Water supply/hydrant	Water pipe	[25]
Innovation flow network	10782	39741	Author	Directed citation link	[26, 27]

Again, we use the ADAM algorithm to solve this problem similar to the optimization of θ.

2.3.3. Optimization of Edge Flow Vector, f. When the edge flow vector is considered, we have the following minimization problem:

$$\min_f \left\{ o_3(f) = \lambda_2 \|\phi - \Phi f\|_F^2 + \lambda_3 \|f\|_F^2 \right\} \quad s.t.\ f_k = \overline{f}_k, \forall k : \varepsilon_k \in E_L. \tag{16}$$

This is a linear constrained quadratic programming problem; we employ the active-set algorithm to solve it.

2.3.4. Optimization of Node Value Vector, ϕ. To solve the node value vector, we fix the other variables and have the following minimization problem:

$$\min_\phi \left\{ o_4(\phi) = \lambda_2 \|\phi - \Phi f\|_F^2 + \lambda_3 \|\phi\|_F^2 \right\}. \tag{17}$$

We set the derivative of o_4 regarding ϕ to zero and have the solution of ϕ as

$$\frac{\partial o_4(\phi)}{\partial \phi} = 2\lambda_2(\phi - \Phi f) + 2\lambda_3 \phi = 0 \implies \phi = \frac{\lambda_2}{\lambda_2 + \lambda_3} \Phi f. \tag{18}$$

2.3.5. Optimization of w. To solve the parameter of function ϕ, we have the following minimization problem:

$$\min_w \left\{ \lambda_1 \sum_{i=1}^{n} \left\| \phi_i - \varphi\left(LSTM\left(x_{i1}, \cdots, x_{i|Ni|}; \theta\right)\right) \right\|_F^2 + \lambda_3 \|w\|_F^2 \right\}. \tag{19}$$

We still employ the ADAM algorithm to solve this minimization problem.

3. Experiments

In this section, we give the experimental setting and results. The algorithm tested and developed is called edge flow prediction by network embedding (EFPNF).

3.1. Benchmark Dataset and Experimental Setting. Four datasets of network are used in our setting. We summarize the datasets in Table 1.

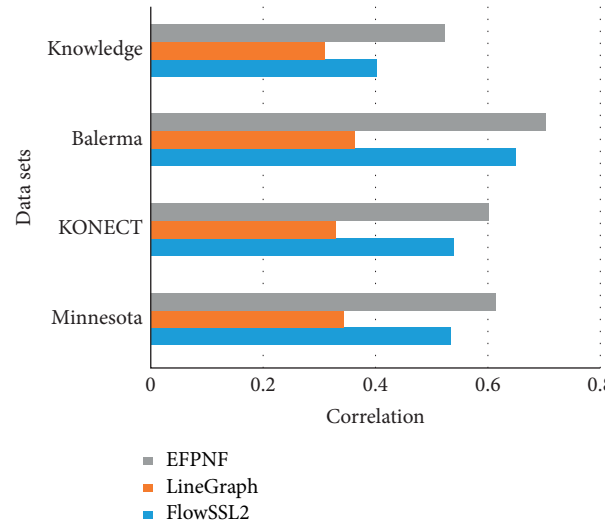

Figure 2: Comparison results over state-of-the-arts.

The 10-fold cross-validation is used to generate the training and test set. The Pearson correlation coefficient is used to measure the quality of the flow predicted by the algorithm [28].

3.2. Experimental Results

3.2.1. Comparison to State-of-the-Arts. As shown in Figure 2, the algorithm is compared to the others which are also used to predict the flows of edges. The compared algorithms are flow SSL algorithm proposed by Jia et al. [1] and the LineGraph algorithm. According to the figure, the proposed algorithm EFPNF has better performance for all four datasets than the compared algorithms. Especially for the Balerma dataset, the EFPNF is the only algorithm which has the correlation score larger than 0.7. We also can see that the knowledge network dataset is the hardest task, and the EFPNF still gives the best performance.

3.2.2. Convergence Analysis. Since our algorithm is an iterative algorithm, we are also interested in the convergence of the algorithm. Thus, we plot the curve of correlation while the number of iterations is increasing (Figure 3). We can see from these curves that our algorithm's performance is boosted when the iteration number is increased from 5 to 20. Since our algorithm aims to minimize the objective function, more iteration numbers result in a smaller value of the objective. This verifies the effectiveness of the objective to

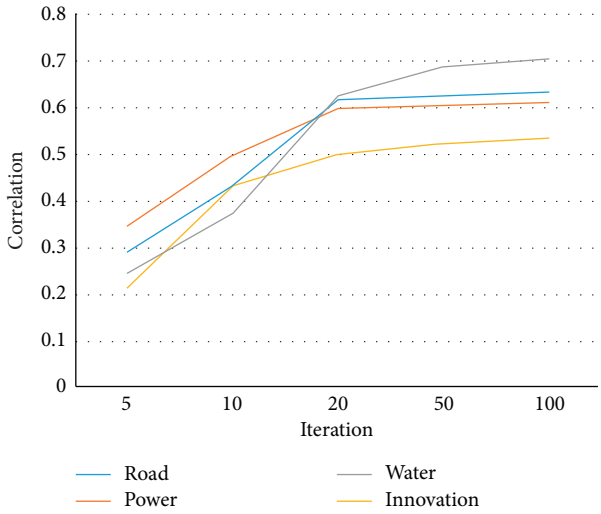

Figure 3: Convergence curve.

achieve a better edge flow estimation performance. However, when the iteration number further increases from 20 to 100, the performance improvement is not significant, meaning our algorithm does not need a large number of iterations to give a good performance.

4. Conclusion

We developed an iterative algorithm to learn the node embeddings and missing flows of the edges for a network. This algorithm is based on the deep recurrent network for the embedding purpose. Moreover, it uses the embeddings to calculate the node values and further uses the node values to approximate the flows around the node. A unified learning framework is built for the learning of embedding network, node value, and flows of the edges. The learning process is guided by the minimization of the reconstruction errors of embedding vectors, node values, and incoming/outgoing flows. Experimental results show the advantage of the proposed algorithm.

Data Availability

All the datasets used in this paper to produce the experimental results are publicly accessed online.

Conflicts of Interest

The authors of this paper claim no conflicts of interest regarding the work reported in this paper.

Acknowledgments

This paper was funded by the National Natural Science Foundation of China (Project nos. 71704036 and 71473062); Social Science Foundation of Ministry of Education of China (Project no. 19YJA790087); and the Talents Plan of Harbin University of Science and Technology: Outstanding Youth Project (Project no. 2019-KYYWF-0216).

References

[1] P. Russom, "Big data analytics," *TDWI Best Practices Report*, vol. 19, no. 4, pp. 1–34, 2011.

[2] M. Chen, S. Mao, and Y. Liu, "Big data: a survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[3] G. Liang, H. Mo, Z. Wang, C. Q. Dong, and J. Y. Wang, "Joint deep recurrent network embedding and edge flow estimation," in *Proceedings of the International Conference on Intelligent Computing*, pp. 467–475, Bari, Italy, October 2020.

[4] E. Alpaydin, *Introduction to Machine Learning*, MIT press, Cambridge, MA, USA, 2020.

[5] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.

[6] X. J. Zhu, *Semi-supervised Learning Literature Survey. Technical Report*, University of Wisconsin-Madison Department of Computer Sciences, Madison, WI, USA, 2005.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[8] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: a survey," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[9] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1975–1981, IEEE, San Francisco, CA, USA, June 2010.

[10] S. Bansod and A. Nandedkar, "Transfer learning for video anomaly detection," *Journal of Intelligent and Fuzzy Systems*, vol. 36, no. 3, pp. 1967–1975, 2019.

[11] T. U. Haque, N. N. Saber, and F. M. Shah, "Sentiment analysis on large scale amazon product reviews," in *Proceedings of the 2018 IEEE International Conference on Innovative Research and Develop- Ment (ICIRD)*, pp. 1–6, IEEE, Bangkok, Thailand, May 2018.

[12] A. Bhatt, A. Patel, H. Chheda, and K. Gawande, "Amazon review classification and senti- ment analysis," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 6, pp. 5107–5110, 2015.

[13] T. N. Sainath, A. R.. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8614–8618, IEEE, Vancouver, Canada, May 2013.

[14] M. Yamada, L. Sigal, M. Raptis, M. Toyoda, Y. Chang, and M. Sugiyama, "Cross-domain matching with squared-loss mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1764–1776, 2015.

[15] G. Zhang, G. Liang, F. Su, F. Qu, and J. Y. Wang, "Cross-domain attribute representa- tion based on convolutional neural network," in *Proceedings of the International Conference on Intelligent Computing*, pp. 134–142, Springer, Wuhan, China, August 2018.

[16] Y. Geng, R. Z. Liang, W. Li et al., "Learning convolutional neural network to maximize pos@ top performance measure," in *Proceedings of the ESANN 2017*, pp. 589–594, Bruges, Belgium, April 2016.

[17] G. Zhang, G. Liang, W. Li et al., "Learning con- volutional ranking-score function by query preference regularization," in *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, pp. 1–8, Springer, Madrid, Spain, November 2017.

[18] Y. Geng, G. Zhang, W. Li et al., "A novel image tag completion method based on convolutional neural transformation," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 539–546, Springer, Sardinia, Italy, September 2017.

[19] K. Wang, J. Liu, and J. Y. Wang, "Learning domain-independent deep representations by mutual information minimization," *Computational Intelligence and Neuroscience*, vol. 2019, Article ID 9414539, 14 pages, 2019.

[20] M. Xiao and Y. Guo, "Feature space independent semi-supervised domain adaptation via kernel matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 54–66, 2014.

[21] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan, "Transferable representation learning with deep adaptation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 12, pp. 3071–3085, 2018.

[22] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 465–479, 2012.

[23] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 2208–2217, Sydney, Australia, August 2017.

[24] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: deep transfer learning through selective joint fine-tuning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1086–1095, Honolulu, HI, USA, July 2017.

[25] T. Suzuki and M. Sugiyama, "Sufficient dimension reduction via squared-loss mutual in- formation estimation," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 804–811, Sardinia, Italy, May 2010.

[26] G. Niu, W. Jitkrittum, B. Dai, H. Hachiya, and M. Sugiyama, "Squared-loss mutual in- formation regularization: a novel information-theoretic approach to semi-supervised learning," in *Proceedings of the International Conference on Machine Learning*, pp. 10–18, Atlanta, USA, June 2013.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends ® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[28] B. Jacob, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, pp. 1–4, Springer, Berlin, Germany, 2009.