C Cara Evangeline
411517 - CSE

SUMMARY

Paper Title: Graph Convolutional Neural Networks for Web-Scale Recommender Systems

Conference: KDD 2018, August 19-23, 2018, London, United Kingdom

INTRODUCTION

Deep learning methods have an increasingly critical role in recommender system applications, being used to learn useful lowdimensional embeddings of images, text, and even individual users.

The main question is, how to define basic deep learning operations for such complex data types. With a growing recommendation service, we don't have the option of a system that won't scale for everyday use. PinSage, a random-walk Graph Convolutional Network is capable of learning embeddings for nodes in web-scale graphs containing billions of objects.

Pinsage shows high-quality embeddings (i.e. dense vector representations) of nodes (e.g., Pins/images) connected into a large graph. The benefit of our approach is that by borrowing information from nearby nodes/Pins the resulting embedding of a node becomes more accurate and more robust.

The core idea behind GCNs is to learn how to iteratively aggregate feature information from local graph neighborhoods using neural networks. Here a single "convolution" operation transforms and aggregates feature information from a node's one-hop graph neighborhood, and by stacking multiple such convolutions information can be propagated across far reaches of a graph. GCNs leverage both content information as well as graph structure. GCN-based methods have set a new standard on countless recommender system benchmarks.

CHALLENGE ADDRESSED

CHALLENGE : The main challenge is to scale both the training as well as inference of GCN-based node embeddings to graphs with billions of nodes and tens of billions of edges. Scaling up GCNs is difficult because many of the core assumptions underlying their design are violated when working in a big data environment. For example, all existing GCN-based recommender systems require operating on the full graph Laplacian during training—an assumption that is infeasible when the underlying graph has billions of nodes and whose structure is constantly evolving.

SOLUTION :

1.On-the-fly convolutions:

Traditional GCN : Graph convolutions by multiplying feature matrices by powers of the full graph Laplacian
Pinsage : Localized convolutions by sampling the neighborhood around a node and dynamically constructing a computation graph from this sampled neighborhood

----->This alleviates the need to operate on the entire graph during training.

2.Efficient MapReduce inference:

Given a fully-trained GCN model, it is still challenging to directly apply the trained model to generate embeddings for all nodes, including those that were not seen during training. Naively computing embeddings for nodes with localized covolutions leads to repeated computations caused by the overlap between *K*-hop neighborhoods of nodes.

We use *map* to project all nodes to the latent space without any duplicated computations, and then *join* to send them to the corresponding upper-level nodes on the hierarchy, and finally *reduce* to perform the aggregation to get the embedding for the upper-level nodes. Our efficient MapReduce-based inference enables generating embeddings for billions of nodes within a few hours on a cluster a few hundred instances.

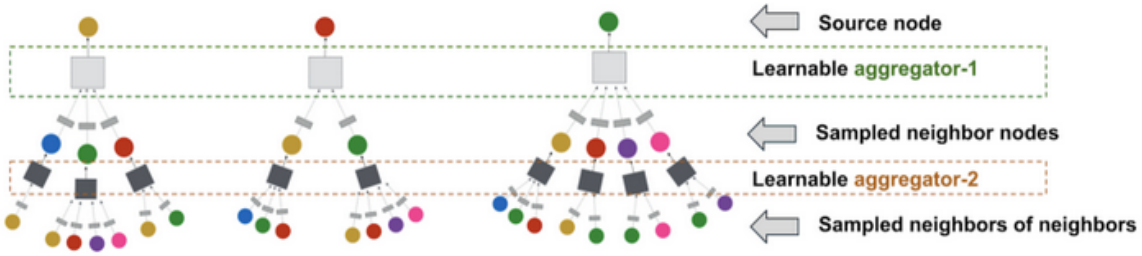

Figure 1: Example of computation graphs we dynamically construct for performing localized graph convolutions. Here we show three source nodes (at the top) for which we are generating embeddings. For each source node, we sample its neighbor nodes and we further sample neighbor nodes of each neighbor, i.e., here depth is 2. Between the layers are learnable aggregators parameterized by neural networks. Aggregators are shared across different computation graphs.

3.Constructing convolutions via random walks

Performing convolutions on full neighborhoods of nodes would result in huge computation graphs, so we resort to sampling. An important innovation in our approach is how we define node neighborhoods, i.e., how we select the set of neighbors to convolve over. Whereas previous GCN approaches simply examine *K*-hop graph neighborhoods, in PinSage we define importance-based neighborhoods by simulating random walks and selecting the neighbors with the highest visit counts.

MODEL ARCHITECTURE

- *Forward propagation algorithm (Algorithm 1)*

The core of our PinSage algorithm is a localized convolution operation, where we learn how to aggregate information from u's neighborhood (Figure 1). The basic idea is that we transform the representations of u's neighbors through a dense neural network. The output of the algorithm is a representation of u that incorporates both information about itself and its local graph neighborhood.

- *Importance-based neighborhoods*

Pinsage simulates random walks starting from node u and compute the L_1 -normalized visit count of nodes visited by the random walk. The neighborhood of u is then defined as the top T nodes with the highest normalized visit counts with respect to node u.

- *Stacking convolutions (Algorithm 2)*

Each time we apply the convolve operation (Algorithm 1) we get a new representation for a node, and we can stack multiple such convolutions on top of each other in order to gain more information about the local graph structure around node u.

EVALUATION

(1) PinSage is implemented on Pinterest data—bipartite Pin-board graph with visual and annotation embeddings as input features. The visual embeddings used are from a state-of-the-art convolutional neural network deployed at Pinterest. Annotation embeddings are trained using a Word2Vec-based production model at Pinterest, where the context of an annotation consists of other annotations that are associated with each Pin. PinSage is evaluated against the following content-based deep learning baselines that generate embeddings of Pins:

- **Visual embeddings (Visual)**: Uses nearest neighbors of deep visual embeddings for recommendations.
- **Annotation embeddings (Annot.)**: Recommends based on nearest neighbors in terms of annotation embeddings .
- **Combined embeddings (Combined)**: Recommends based on concatenating above visual and annotation embeddings, and using a 2-layer multi-layer perceptron to compute embeddings that capture both visual and annotation features. It is trained with the exact same data and loss function as PinSage.

(2) The effectiveness of PinSage is also observed by performing head-to-head comparison between different learned representations. In the study, a user is presented with an image of the query Pin, together with two Pins retrieved by two different recommendation algorithms. The user is then asked to choose which of the two candidate Pins, if either, is more related to the query Pin. Users are instructed to find various correlations between the recommended items and the query item, in aspects such as visual appearance, object category and personal identity. If both recommended items seem equally related, users have

the option to choose “equal”. If no consensus is reached among 2/3 of users who rate the same question, we deem the result as inconclusive.

CONCLUSION

PinSage, a random-walk Graph Convolutional Network that is highly-scalable and capable of learning embeddings for nodes in web-scale graphs containing billions of objects was proposed to

- Recommend related pins and
- Recommend pins in a user’s home/news feed.