Topics and Tools on Social Media Data Mining

CS529

Assignment 3

Evaluation Report on

Exploration of Multi-View Network Representation Learning

Bhupender (194101014)

Vandana Mishra (194101055)

Mayank Verma (194101031)

IIT GUWAHATI

Introduction:

Most of the existing works consider homogeneous network with single type of node and single type of relations. Based on which, we generate some useful results like node classification, link prediction, and node visualization.

Problem: Most of the real-world network information networks are heterogeneous in nature, I.e. having multiple types of nodes and relations. This will give rise to different views of the same network.

Multiple relations yield a network with multiple views.

We generally do all these prediction and classifications on some numerical data, not the original node and link descriptions.

So, Vector Representation of nodes need to be carried out. But as each individual view is usually sparse and biased. Hense the node representations learned by existing approaches may not be comprehensive.

Problem Statement:

Primary Objective: To learn robust node representation by considering multiple views of a network.

Task 1: To perform node embedding for each view i.e. K vectors for each node, using

- Neural network-based technique.

Task 2: To perform node embedding by collaborating all the views i.e. a single vector for each node, using

- Multi-view matrix factorization techniques

Task 3: To apply the embedding obtained above for the Link Prediction task.

Task 1: Obtain Node Embeddings for each view.

We are using Node2vec approach to find the features / vector representation of the nodes in the network for each view.

1. Why node2vec

Any supervised machine learning algorithm requires a set of informative, discriminating, and independent features. However, classic approaches based on linear, non-linear dimensional reduction like PCA and others are much expensive in term of time and computationally for large real-world networks, which also are heterogeneous. Different experiments on node2vec demonstrate state-of-the-art performance. It gives better results up to 26% on multi-labeled classification and 12% on link prediction.

2. Approach

The proposed node2vec, a semi-supervised algorithm for feature learning in network. They optimize a graph-based objective function using SGD. They use 2^{nd} order random walk to generate network neighborhoods for nodes.

3. Feature learning framework

They formulate this as maximum likelihood optimization problem. Let G = (V, E) be a given network. Can be apply any graph. Let f: V -> R^d be mapping function from node to feature representation we aim to learn. Here d is a parameter specifying number of dimensions of feature space. And f is matrix of size |V| * d. Optimize the following objective function, which maximizes the log-probability of observing a network neighborhood Ns(u) for a node u, conditioned on its feature representation, given by f:

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u)). \qquad (1)$$

In order to make optimization problem tractable, two assumption were made:

-> Conditional independence. Neighborhood of two different node is not depend on each other.

-> Symmetry in feature space. A source node and neighborhood have a symmetric effect over each other in feature space.

The neighborhoods Ns(u) are not restricted to just immediate neighbors but can have vastly different structure depending on sampling strategies.

3.1. Sampling search strategies

-> Breadth-first Sampling (BFS). The neighborhood ns is restricted to nodes which are immediate neighbors of the source.

-> Depth-first Sampling (DFS). The neighborhood consists of nodes sequentially sampled at increasing distance from source node.

3.2 node2vec

The aim is to make a flexible neighborhood sampling strategy which allow to smoothly interpolate between BFS and DFS.

3.2.1 Random walks

Given a source node u, random walk simulates for fixed length l. Let ci denote the ith node in walk, starting from c0= u. Nodes ci are generated by following distribution:

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

Where πvx is the transition probability between nodes v and x, and Z is normalizing constant.

3.2.2 Search bias α (2nd order Random walk)

This is the main strategy which help to interpolate between BFS and DFS smoothly. This is defined with two parameters p and q, which guide the walk: Consider a random walk that just traversed edge (t, v) now reside at node v. The walk needs to decide on the next step, so it evaluate the transition probabilities πvx on edge (v, x) leading from v. We set the unnormalized transition probability to πvx = αpq(t, x) * w(vx), where

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

And dtx denotes the shortest path distance between node t & x. dtx = {0,1,2} only.

-> Return parameter, p. High value of p ensure that next node is not visited. Lower value of p leads to next node is already visited.

-> In-Out parameter, q. If q > 1 walk strategy will BFS. And if q < 1 walk strategy will DFS.

3.2.3 The node2vec Algorithm

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
π = PreprocessModifiedWeights(G, p, q)
$G' = (V, E, \pi)$
Initialize $walks$ to Empty
for $iter = 1$ **to** r **do**
  for all nodes $u \in V$ **do**
    $walk$ = node2vecWalk(G', u, l)
    Append $walk$ to $walks$
f = StochasticGradientDescent(k, d, $walks$)
return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
Inititalize $walk$ to $[u]$
for $walk\_iter = 1$ **to** l **do**
  $curr = walk[-1]$
  V_{curr} = GetNeighbors($curr, G'$)
  s = AliasSample(V_{curr}, π)
  Append s to $walk$
return $walk$

i. graph: The first positional argument has to be a networkx graph. Node names must be all integers or all strings. On the output model they will always be strings.
ii. dimensions: Embedding dimensions (default: 128)
iii. walk_length: Number of nodes in each walk (default: 80)
iv. num_walks: Number of walks per node (default: 10)
v. p: Return hyper parameter (default: 1)
vi. q: Inout parameter (default: 1)

Applying node2vec for obtaining embeddings for all the views separately.

Task 2: To perform node embedding by collaborating all the views.

We are using Matrix factorization approach for integrating multiple data views in given dataset.

1. Why Integration by Matrix Factorization (IMF)

This is unsupervised algorithm for combining information from related views, using a late integration strategy. This take representative clustering generated independently on each available view and applies a factorization procedure to this representation to reconcile the groups arising from the individual views.

For some data exploration applications, we may have access to a set of views that are entirely compatible– the same patterns will occur across all views. The problem then becomes the identification of a single consensus model describing the patterns common to all views. In other cases, significant discord may exist between the data in different views. An effective data integration procedure must then reconcile these disagreements, identifying common patterns, while also preserving those that are unique to each view.

The fact that IMF operates on previously generated clusterings alone, rather than any specific representation of the original data, neatly avoids the diversity of representation issue.

2. Approach

The basic strategy is late integration.

There are basic three integration strategies, ***Early integration*** involves direct combination of data from several views into a single dataset before learning. ***Intermediate integration*** involves computing separate similarity matrices on the views and producing a combined pairwise representation which is then passed to learning, and ***Late integration*** involves applying an algorithm to each individual view and subsequently combining the results.

2.1. Intermediate representation

Given a set of V views {v1, v2, …. ,vV} with total n nodes in domain. And we have clustering of each view {c1, c2, …...., cV}, where cH indicate the set of clusters {cH1, … ,cHk}. And l is total number of clusters in all views.

All these clusters are representation by set of non-negative membership matrices M = {m1, m2, …, mV}, where mH belongs to R (n * kH). By transposing membership matrices and stacking all of them vertically, this gives us X (l * n) as embedding of the original object in l-dimensional space.

2.2. Factorization process

The goal of integration process is to project the X in low dimensional space. For this X is decomposed into two factor **P** and **H** where both are non-negative. **P** (l * k') and **H** (k' * n).

Task 3: Apply link prediction task on above obtained embeddings.

1. Results:

Classifier Used: Naïve Bayes

1.1 For individual views:

	Precision	Recall	AUC score	F-score
View0	0.833079	0.831178	0.87075	0.876226
View1	0.881969	0.778221	0.836875	0.826854
View2	0.919599	0.892768	0.907125	0.905985
View3	0.828196	0.765230	0.801375	0.795469
View4	0.896756	0.804558	0.8515	0.848159
View5	0.933096	0.911837	0.9225	0.922344
View6	0.876349	0.816603	0.851625	0.845422
View7	0.866127	0.821331	0.845625	0.843134
View8	0.842105	0.882205	0.85875	0.861689

1.2 For average view:

	Precision	Recall	AUC score	F-score
Average View	0.747099	0.728506	0.742375	0.737686

1.3 For complete view, after applying multi-view matrix factorization:

	Precision	Recall	AUC score	F-score
Collaborated View	0.796088	0.804347	0.79675	0.800196

Classifier Used: Random Forest Classifier (n_estimators=30)

2.1 For individual views

	Precision	Recall	AUC score	F-score
View0	0.928321	0.831178	0.86775	0.875412
View1	0.875412	0.933568	0.901	0.903555
View2	0.930851	0.960339	0.944375	0.945365
View3	0.847036	0.909228	0.87325	0.877031
View4	0.892445	0.950852	0.918375	0.920723
View5	0.926728	0.955332	0.940125	0.940813
View6	0.957029	0.973225	0.965125	0.965059
View7	0.961453	0.982685	0.97175	0.971953
View8	0.873762	0.907455	0.89125	0.890290

2.2 For average view

	Precision	Recall	AUC score	F-score
Average View	0.715296	0.851585	0.754125	0.777513

2.3 For complete view, after applying multi-view matrix factorization:

	Precision	Recall	AUC score	F-score
Collaborated View	0.833079	0.831178	0.834625	0.832127

Conclusion

1. Individual views give high accuracy, when used for link prediction. But it is not accurate as individual views can't predict for the whole network.
2. Matrix factorized collaborated view give good accuracy for link prediction than that of simple average view of the whole network.

References

Repository: https://github.com/aditya-grover/node2vec

https://github.com/yifeng-li/mvmf

Papers: https://arxiv.org/pdf/1607.00653.pdf

http://derekgreene.com/papers/greene09ecml.pdf