Cross Validation on Graphs

This semester, I had the unique oppertunity to work as an undergrad researcher at a machine learning lab at Rutgers. One of my duties was to perform 10-fold cross validation on some baseline machine learning models to compare our model to after it had been created. My steps included preprocessing data, splitting into buckets, generating embeddings for each bucket, and calculating the ROC-AUC value after each iteration of the cross validation, finally I had to calculate the mean and standard deviation of the 10 ROC-AUC values.

The Problem

The researchers at the Artificial Visual Interfaces Lab are trying to create a new machine learning model to compress large graphs into something called embeddings, at the same time we don't want to lose certain properties of graphs. I was tasked with finding a way to compare our newly created model with previously existing ones (from other research labs) to see how much information our compression retained.

How did I solve the problem

I ran cross validation techniques on three previously existing models so that we can compare how well the new model does (the links can be found below). The property which I examined for each model is called Link prediction (a form of classification), basically, certain edges in the graph were removed before the embedding layer was generated so the model had never encountered these edges, the **removed edges** are the **testing set**. The **training set** are the edges **leftover** after the testing edges were removed. I also had to convert the training sets into acyclic graphs, I used a method called pagerank for this process. This was basically the preprocessing portion of the cross validation.

```
break_cycles_to_DAG(s.join(myFile), s.join(myDeletedEdges))
break_cycles(s.join(myFile), s.join(myDeletedEdges), None, "pagerank", int)
```

Because graphs are disjoint entities, the cross validation was modified. Each fold had its own training and testing set. A structure of my test data can be found below. Thus, although 10 runs were calculated, the values were not the same.

```
├── 10 Fold Cross Validation
│   ├── Fold 1
│   │   ├── Embedding (My Model)
│   │   ├── Training Graph
│   │   ├── Testing Graph (Taken randomly from the original graph)
│   ├── Fold 2
│   │   ├── Embedding (My Model)
│   │   ├── Training Graph
│   │   ├── Testing Graph (Taken randomly from the original graph)
|   |   ...
│   ├── Fold 10
│   │   ├── Embedding (My Model)
│   │   ├── Training Graph
│   │   ├── Testing Graph (Taken randomly from the original graph)
```

Three versions of this folder structure were created, a different one for each Embedding (Model). Once I had generated my three different models, it was time to test them. I did this by using a technique called link prediction which is a classification technique. Basically the model generated vectors for each node, and the likelihood of an edge can be calculated by taking the dot product of the two vectors. The observed result was compared to the expected result, and we have a value.

```
for e in pos_edges:
    preds_pos.append(sigmoid(numpy.dot(hub[e[0]], auth[e[1]].T)))
labels_all = numpy.hstack([numpy.ones(len(preds_pos))])
metrics.roc_auc_score(labels_all, preds_all)
```

My Results

Method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	Mean	Std Dev
NERD	0.7692	0.6951	0.5975	0.6919	0.7048	0.6872	0.7279	0.736	0.7319	0.6805	0.7022	0.0459
HGCN	0.6328	0.6309	0.6341	0.6130	0.6313	0.6412	0.6445	0.6205	0.6453	0.6330	0.6327	0.0101
GGCN	0.6948	0.7022	0.6931	0.6926	0.689	0.6967	0.6955	0.6776	0.6984	0.687	0.6927	0.0068

What does it mean?

Although the standard deviation of all three models was low, the best results were seen with NERD. This means that the model created by the NERD code was able to predict relationships between two nodes better than the other two models. Of course, results may vary based on the data itself. Maybe some of the models work better on graphs without cycles and and some work better on undirected graphs, the only way to know would be to find other datasets which have different properties and run the models on them.[1]

Next Steps

Because the way the embeddings are generated, there is a degree of randomness which can only be measured by repeated k-fold cross validation. Therefore in the next month, I will run the 10-fold cross validation 10 times for these 3 embedding techniques and find the average and standard deviation of each cross validation run and average that at the end. I will also start calculating Average Precision scores along with ROC-AUC scores, Average Precision is another well known classification score and in this situation we use it to see how well our embeddings can prodict links. In adition, I will be carrying out bi-directionality prediction testing and do repeated 10-fold cross validation on those seperate datasets as well. In addition I will be testing both link prediction and bi-directionality prediction on two other data sets with distinct properties to see how the models work.

What concepts did I use?

There are so many ways statistics is applied to different fields. Obviously k-fold cross validation was a big part in my research, and repeated k-fold cross validation makes this process even more time consuming. The concept of a sigmoid though it is applied differently in machine learning is also a concept that I used, the fact that it takes a value and maps it between 0 and 1. Also scores such as average precision and roc auc were concepts that I first saw in class when we were learning logistic regression, I was surprised to learn that they could be used in any type of classification technique, not just logistic regression.

Who helped me out?

My researcher (a PHd student at Rutgers) instructed me on how to calculate the ROC - AUC statistic, otherwise the cross validation process was completed by myself. The papers linked to throughout this file also helped me a good amount because I saw how those did their cross validation.

Whose work did I refer to?

[1]GNN