Winter 2019 UCLA CS249 Project Report Paper Classification on Citation Networks by Using Asymmetry Relation Design

Group 2 Pokemon

Zixiang Liu
zliu46@g.ucla.edu

Tianyi Ma
timtianyima@gmail.com

Enbang Zhang
zhangenbang1996@gmail.com

Chengyao Zhang
cyzhang2018@ucla.edu

Jiachen Zhong
julightzhong10@cs.ucla.edu

ABSTRACT

In this project, we designed our own Probabilistic Graphical Model to perform paper topic classification by utilizing citation information. We applied two types of potential functions in order to compare the difference between symmetrical and asymmetrical citation relations. For performing a better parameters learning, several different methods were implemented, compared, and analyzed. In addition, we carefully designed methods to split test and training set in order to perform meaningful validation. Moreover, we implemented a Gibbs Sampling based method to perform inference when multiple nodes in our graphs need to be predicted. Finally, we found utilizing asymmetrical citation relations dose improve the classification result and we provide some analyses based on our understanding.

1. INTRODUCTION

An academic paper is usually associated with a research topic and is related to other papers in citation relations. As we all know, the topic of a paper is highly related to the paper it cites as well as those citing it. They tend to be on the same topics and maybe several topics of papers are more likely to cite each other. Therefore, the goal of our project is to predict the topic of papers topic according to the paper cited and citing by it and its title. We used Probabilistic Graphical Model as the model and tried to interpret the parameter to explain the relationship between each topic.

Besides, the citing and cited relationship should be different. For instance, if a paper is about neural network, it should be more likely to cite but not to be cited by a Theory paper. To get the asymmetric information and compare it to the model which only take symmetric information into account, we designed both asymmetric and symmetric potential functions and tested their performance.

© 2019 ACM. ISBN 123-4567-24-567/08/06.
DOI: 10.475/123_4

2. RELATED WORKS

2.1 Probabilistic graphical models

A Probabilistic Graphical Model(PGM) is probabilistic model used to describe the dependency between random variables in a graph. A PGM is usually applied in relational data cases because it is suitable to describe relations among data. There are many types of PGMs like Hidden Markov Model (HMM) [12], Markov Random Field(MRF) [3], Conditional Random Field (CRF) [6], and Bayesian Network [2]. Among those PGMs, there are directed and undirected types which process relations between data as asymmetrical and symmetrical. For example, MRF and CRF are usually used as undirected PGM, while Bayesian Network is directed model. Basically, different PGMs can be unified as different Factor Graphs [5] holding different factor nodes or potential functions. Specially designed potential function can be used to describe different type of relations. And in this project, we mainly used MRF with specially designed symmetrical and asymmetrical potential function to represent the citations network.

A MRF is a probability distribution P over random variables $x_1, x_2, , x_n$ defined by a graph G.

$$P(x_1, x_2, ..., x_n) = \frac{\prod_{c \in C} \phi_C(x_c)}{Z} \tag{1}$$

where $\phi_c(\cdot)$ is the potential function, c is clique belonging to the whole collection of cliques C, and Z is the partition value for normalization.

2.2 CORA dataset and previous works

In our project, we used the CORA [13] dataset. It consists of 2708 scientific publications in the machine learning area. Those papers are classified into one of seven classes which are Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory. For each paper, the dataset has the paper cited by it and citing it. The citation network consists of 5429 links. Besides, the dataset has the title of each paper so we can utilize both kinds of information. CORA is a very classical dataset, and many works have used this relational dataset to verify several different methods. In 2000 and 2003, Jennifer Neville and David Jensen' team used Bayesian classifiers and relational probability trees to perform the classification [8, 10]. More related to our project,

there are several researches in collective classification approaches, which utilize the dependencies between data, to perform the benchmarks [9, 1, 7]. More recently, graph embedding method [11] and graph convolutional networks [4] were also tested on this dataset.

In this project, although we did the above background research, we did not borrow any of specific algorithm above. Since we had limited time and our main goal is to compare the effects between symmetrical and asymmetrical relation design in this dataset, we deigned our own models with relatively simple structure. The detail of the design can be found in Section 3.

3. METHODS

3.1 Model design

3.1.1 Model and potential function

In this design, we represented the symmetry and asymmetry citation relations in potential function. We firstly define notations. We use n_i denotes node i and N_i denotes all the neighbours of n_i. There are two types of neighbours which citing the n_i or are cited by n_i. We represent citing as $\oplus$ and cited as $\ominus$. Therefore, $N_i^{\oplus}$ are the neighbours citing n_i and $N_i^{\ominus}$ is are the neighbours cited by n_i. And $N_i = N_i^{\oplus} \cup N_i^{\ominus}$. We use x_i denotes the paper class and t_i denotes the text information of of n_i. After setting notation, the whole model which is a joint probability is:

$$P(x_0, ..., x_n) = \frac{\prod_i T(t_i)\phi(n_i, N_i)}{Z} \tag{2}$$

where t_i is the title information of n_i, $T(\cdot)$ is the pretrained naive bayes classifier to give the probability of x_i to n_i according to t_i, $\phi(\cdot)$ is the relation potential function, and Z is the partition value. For single node, the probability of n_i belongs to x_i only depends on its neighbours N_i and its title t_i. This can be represented in the following formula:

$$P(x_i) = \frac{T(t_i)\phi(n_i, N_i)}{Z'} \tag{3}$$

For the potential function, we first introduce the general relation potential function is defined as:

$$\phi(n_i, N_i) = \frac{\frac{\lambda}{k} + \sum_{j \in N_i} I(x_i, x_j)R(n_i, n_j)}{|N_i| + \lambda} \tag{4}$$

where $|N_i|$ is the number of n_i's neighbours, λ is a positive smoothing factor to avoid empty neighbour case, k is the number of paper classes, $R(\cdot)$ is a function which depends on the relations of two nodes and returns value, $I(\cdot)$ is the identity function which will return 1 when $x_i = x_j$ otherwise 0. The meaning of $\phi(\cdot)$ is to calculate a value to represent how confident n_i belonging to a class according to its neighbours. Therefore, the symmetry and asymmetry citation relations will be represented in different function $R(\cdot)$. For symmetry citation relations:

$$R(n_i, n_j) = \theta_{x_i x_j} \tag{5}$$

where $\theta_{x_i x_j}$ is the weight parameter when n_i is class x_i and n_j is class x_j. In total, we have k^2 parameters in the model under symmetry case. This relation function is symmetrical because it dose not consider the citing and cited cases. Therefore, in order to represent asymmetrical relations in paper, citing and cited cases need to be involved:

$$R(n_i, n_j) = \begin{cases} \theta^{\oplus}_{x_i x_j} & when \quad n_i \oplus n_j \\ \theta^{\ominus}_{x_i x_j} & when \quad n_i \ominus n_j \end{cases} \tag{6}$$

where $\theta^{\oplus}_{x_i x_j}$ is the weight parameter when n_i is class x_i, n_j is class x_j, and $n_i \oplus n_j$; while $\theta^{\ominus}_{x_i x_j}$ is the weight parameter when n_i is class x_i, n_j is class x_j, and $n_i \ominus n_j$. In total, we have $k^2 \times 2$ parameters in the model under asymmetry case.

In both symmetry and asymmetry cases, all θ can be manually picked or automatically learned. For the learning process, we hope all the θ is positive because if $\phi(\cdot)$ returns a negative value it will effect the $T(\cdot)$ which is always a positive probability. Thus, we need to make constrains to make sure the learned θ is positive. The detail will be discussed in the following training section.

3.1.2 Loss-based training

Training the weights is a very critical process for prediction. However, an universal approach which works for all sorts of graphical model is never at our disposal. Therefore, we need to define an unique approach which can bring performance improvement to our designed potential function. But note that the approach is not limited to the one this paper proposes, and as will be discussed in the later section, there are some other potential methods worth considering as well.

Look back to the identity function $I(x_i, x_j)$, which can be easily converted to a feature vector v_i for each node N_i.

$$v_i = (C_i(\oplus, x_1), C_i(\ominus, x_1), ..., C_i(\oplus, x_k), C_i(\ominus, x_k)) \tag{7}$$

where $C_i(\oplus, x_i)$ is the count of N_i's positive neighbours of class x_i and $C_i(\ominus, x_i)$ is the count of N_I's negative neighbours of class x_i. And therefore we converted every node to a feature vector of length $|v_i| = 2k$, where k is number of classes.

We utilized the same arrangement for weight parameters and achieved a $(k \times 2k)$ matrix W:

$$\begin{bmatrix} \theta^{\oplus}_{x_1 x_1} & \theta^{\ominus}_{x_1 x_1} & \cdots & \theta^{\ominus}_{x_1 x_k} \\ \theta^{\oplus}_{x_2 x_1} & \theta^{\ominus}_{x_2 x_1} & \cdots & \theta^{\ominus}_{x_2 x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \theta^{\oplus}_{x_k x_1} & \theta^{\ominus}_{x_k x_1} & \cdots & \theta^{\ominus}_{x_k x_k} \end{bmatrix}$$

Each entry in the matrix W represents the strength of an association between two classes. For example, $\theta^{\oplus}_{x_1 x_1}$ reflects the strength of association that a class-1 paper also cites a class-1 paper, while $\theta^{\ominus}_{x_1 x_1}$ reflects the strength association that a class-1 paper is cited by another class-1 paper. As multiply the feature vector vi with the weight matrix W, the dot product is an array of scores indicating N_i's class-membership for each class, respectively.

$$\begin{aligned} v_i W^T &= [s_1, s_2, ...s_k] \\ s_1 &= C_i(\oplus, x_1)\theta^{\oplus}_{x_1 x_1} + ... + C_i(\ominus, x_k)\theta^{\ominus}_{x_1 x_k} \end{aligned} \tag{8}$$

But it is hard to deal with scores with no boundary; for an observed paper, there is no limit on how high the score can be assigned to the correct class and how low the score can be assigned to the incorrect ones. Thus, we normalized the score and re-interpreted them as the probability of a paper

belonging to a certain class.

$$\hat{y_i} = \frac{s_i}{\sum_{n=1}^{k} s_n} \tag{9}$$

Then by assigning value 1 to the probability of the correct class and value 0 to the rest classes, we introduced our loss function $J(\hat{y})$ as following:

$$J(\hat{y}) = \frac{\sum_{i=1}^{k}(\hat{y_i} - y_i)^2}{k} \tag{10}$$

After several conversions, the goal of learning becomes clear and that is learning weight matrix W such that loss function J is minimized. Define our input matrix x as:

$$x = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \tag{11}$$

The weight matrix W is initialized with random number distributed uniformly between 0 to 1, and we decided to update the matrix with **GradientDescent**. Since the the normalization is involved, we separated the loss function with each class for convenience of understanding and computation. And for a certain class i, its update scheme for the corresponding row in the weight matrix is shown below:

$$\begin{aligned} J_i(\hat{y}) &= (\hat{y_i} - y_i)^2 \\ \hat{y_i} &= \frac{xW_i^T}{Z} \\ Z &= \sum_{n=1}^{k} xW_n^T \end{aligned} \tag{12}$$

The weight matrix W was updated each iteration but constrained by a small learning rate η. Adding any constant coefficient could only apply a very small effect on the final result. To simplify the computing process, all constant coefficients were dropped automatically in the follow equations:

$$\begin{aligned} W_{i\ new}^T &= W_{i\ old}^T - \eta \frac{\partial J_i}{\partial W_{i\ old}^T} \\ \frac{\partial J_i}{\partial W_{i\ old}^T} &= \frac{\partial J_i}{\partial y_i}\frac{\partial y_i}{\partial W_{i\ old}^T} = (\hat{y_i} - y_i)\frac{x(Z - xW_i^T)}{Z^2} \end{aligned} \tag{13}$$

As expected, above setting worked well for the CORA data set, but it had a potential problem that the weight learned from this method could be negative. The negative number should be prohibited in the sense that normalization process requires every input to be positive; otherwise, it will generate negative probability which does not make any sense. In addition, negative weight should be avoid according to the semantics of the weight. As analyzed in the previous paragraphs, the weights indicate the strength of association and therefore should never be less than zero. In order to solve this problem, we chose to use the square of each weight to calculate the score. Even though it is not the only solution, we find this scheme easy to implement and, in particular, effective enough to generate good results without complicating the model too much. Now with the square function, a slightly different loss function and gradient is shown below:

$$\begin{aligned} \hat{y_i} &= \frac{xW_i^{T2}}{Z} \\ Z &= \sum_{n=1}^{k} xW_n^{T2} \\ \frac{\partial J_i}{\partial W_i^T} &= (\hat{y_i} - y_i)\frac{xW_i^T(Z - xW_i^{T2})}{Z^2} \end{aligned} \tag{14}$$

Similar results can be achieved using softmax:

$$\begin{aligned} \hat{y_i} &= \frac{e^{xW_i^T}}{Z} \\ Z &= \sum_{n=1}^{k} e^{xW_n^T} \\ \frac{\partial J_i}{\partial W_i^T} &= x^T(\hat{y_i} - y_i)(\hat{y_i} - \hat{y_i}^2) \end{aligned} \tag{15}$$

3.1.3 Alternative schemes

Alternative methods are also worth considering, even though due to the limit of time they are not put into the agenda of experiment. The reason they are discussed here is to extend the scope of this paper and lay the groundwork for future work.

Sigmoid function is a solid candidate to probability function. Since the range of sigmoid function is fixed at $(0, 1)$, the concern against negative values should not haunt us any more. In this case the output vector should be:

$$\hat{y_i} = \frac{e^{xW_i^T}}{e^{xW_i^T} + 1} \tag{16}$$

In essence, it then becomes a logistic regression task, which will perform poorly on linearly inseparable data set. But after all, the overall model is trying to capture the relation within the data set and there is no sign indicating that a linear separator should fail to serve as a potential function.

Another angel to approach this question is from the probabilistic aspect of citing a paper; to learn the potential function is through modeling the activity of citing and being cited as a process of multinomial distribution. Let p_{ij} represents the probability that paper from class i citing a paper from class j. Assume the citation of each paper is independent on one another and probability of a paper from class i citing a paper from class j is equivalent to the probability of a paper from class j being cited by a paper from class i, then the probability of a single node N_i's neighbours is:

$$\begin{aligned} P(\oplus N_i) &= \frac{\oplus n_i!}{x_1!x_2!...x_k!} p_{i1}^{x_1} p_{i2}^{x_2} ... p_{ik}^{x_k} \\ P(\ominus N_i) &= \frac{\ominus n_i!}{x_1!x_2!...x_k!} p_{1i}^{x_1} p_{2i}^{x_2} ... p_{ki}^{x_k} \end{aligned} \tag{17}$$

where N_i has $\oplus n_i$ positive neighbors and $\ominus n_i$ negative neighbors, and x_k represents the number of positive($\oplus N_i$) or negative($\ominus N_i$) neighbors of class k. Then the probability of observed graph is:

$$P(G) = \sum_{N_i \in G} P(\oplus N_i) P(\ominus N_i) \tag{18}$$

And apparently the learning process is to find the parameters p_{ij} that maximize the overall probability of the graph.

For this sake, we can harness the MLE to estimate each p_ij. Taking this model even further, we can also introduce prior probability of each class and devise a EM algorithm to learn all the parameters.

These are all paths we can jump onto in the future, nonetheless, neither of them is guaranteed to produce a better results than our current models. Therefore, we shall proceed with cation.

3.2 Pseudo-likelihood and gradient ascent

For an MRF with parameters θ,

$$P(x_1, ..., x_n|\theta) = \frac{1}{Z(\theta)} \prod_c \phi_c(x_c|\theta) \tag{19}$$

The c here is the clique which is basically each edge of the graph. $\phi_{ij}(x_i = m, x_j = n) = exp(R_{mn})$ Our goal is to maximize this likelihood function(maximum likelihood estimation). However, it is intractable to compute the gradient so we cannot directly use gradient ascent. Therefore, instead of maximizing the likelihood function directly, we maximize the pseudo-likelihood which is approximately equal to the likelihood function but whose gradient is very easy to compute.

$$pseudo - likelihood = \prod_i P(x_i|x_{N(i)}; R) \tag{20}$$

It is an intuitive way to compute the likelihood if we only take the neighbors into consideration. The pseudo-likelihood is also concave so we can guarantee to get the global optimal.

$$P(x_i|x_{N(i)}; R) = \frac{T(t_i)e^{\sum_{j \in N(i)} \sum_{m,n} 1(x_i=m, x_j=n) R_{mn}}}{Z_i(R)} \tag{21}$$

$$\frac{\partial log(p(x_i|x_{N(i)}; R))}{\partial R_{mn}} = \sum_{j \in N(i)} 1(x_i = m, x_j = n) - E_{x_i|x_{N(i)}} \sum_{j \in N(i)} 1(x_i=m, x_j=n) \tag{22}$$

The whole gradient for the pseudo-likelihood function is the sum for the gradient of this function for all the nodes because the sum of this function is the pseudo-likelihood function. There are also some trick to do the gradient ascent in practice. First, the learning rate should be small enough to avoid the case the gradient makes $p(x_i|x_{N(i)}; R)$ infinite. Second, the theta should be updated once after all the gradients have been computed or the theta needs to computes the number of papers times in 1 iteration which will be a waste of time.

4. EXPERIMENTS AND EVALUATION

4.1 Experiment data set

The dataset we used in this project is CORA dataset. The CORA dataset consists of Machine Learning papers. These papers are classified into one of the following seven classes: Case_Based, Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, Theory. The papers were selected in a way such that in the final corpus every paper cites or is cited by at least one other paper. There are 2708 papers in the whole corpus.

After stemming and removing stop words we were left with a vocabulary of size 1433 unique words. All words with document frequency less than 10 were removed.
There are two input files:
The cora.content file contains descriptions of the papers in the following format:

$$< paper\_id >< word\_attributes > + < class\_label >$$

The first entry in each line contains the unique string ID of the paper followed by binary values indicating whether each word in the vocabulary is present (indicated by 1) or absent (indicated by 0) in the paper. Finally, the last entry in the line contains the class label of the paper.

The cora.cites file contains the citation graph of the corpus. Each line describes a link in the following format:

<ID of cited paper> <ID of citing paper>

Each line contains two paper IDs. The first entry is the ID of the paper being cited and the second ID stands for the paper which contains the citation. The direction of the link is from right to left. If a line is represented by "paper1 paper2" then the link is "paper2→paper1".

4.2 Splitting Training and Testing set

However, the original graph is not a connected graph. There are 78 connected components (not necessarily strongly connected components) with the largest one of size 2485. Since the algorithm proposed in this paper requires the graph to be connected, otherwise, the independency of different connected components will make it meaningless in our learning, we only use the 2485 nodes as our dataset, ignoring all the nodes that are not in the largest connected component.

Among these 2485 nodes, we needed to cut the graph into two connected components without breaking the original link structure within each connected component. We applied the following algorithm to accomplish the data preprocessing task (the detailed implementation can be seen in the cutter.py file):

1. Construct the original input graph according to cora.cites file.
2. Using the union-find algorithm to group all the nodes, i.e., find all the connected components.
3. Remove all the small connected components
4. Select a node as the starting node (in our experiment, it's 56112) and apply BFS to this graph. Each time we visit a node, remove all the incident edges of this node. Stop when the number of visited nodes reach the threshold (in our experiment, it's 200).
5. Apply similar algorithm as in step 2 to the remaining and find the largest connected component.
6. Construct the training graph using the largest connected component.
7. Construct the test graph using the remaining graph after cutting the largest connected component from the original graph.

In our experiment, the number of nodes in the training graph is 2191 and the number of nodes in the test graph is 294. The training accuracy of asymmetric prediction is 93.2048%, and the training accuracy of symmetric prediction is 93.2966%. Using asymmetric potential function, the test accuracy is 78.81% and using symmetric potential function, the test accuracy is 79.62%.

Table 1: Symmetric vs Asymmetric accuracy on test set

Accuracy	Symmetric	Asymmetric	Asymmetric Improvement
Pure Naive Bays	73.27	73.27	+0.00
Learning with negative theta	86.18	**87.10**	+0.92
Square Loss function	86.18	**87.56**	+1.38
Softmax Loss function	87.56	**88.02**	+0.46
Pseudo Likelihood	**91.71**	91.24	-0.47
Manual Label	86.18	**87.10**	+0.92

4.3 Prediction

The query process of our model requires both the knowledge of unary information, in our case the word count, and the class labels of all neighbors. If the class labels of neighbor is unknown, the query becomes a simple non-graphical model classification problem based on word count representation.

The split of train and test set ensures that test set nodes are connected to each other. Therefore, it is hard to directly query the class of each node in test set as many of their neighbors does not have a label yet. To address this problem, we designed a prediction algorithm to query the whole test set inspired by Gibbs Sampling.

Let V be the set of nodes in test set

Let E be the set of edges between test and train set

Let G be the graph generated by training nodes and edges

Data: Training and Testing set
Result: An assignment of class to all Testing nodes
Initialization:
V:$\{v_1, v_2, ..., v_n\}$,
E:$\{e_1, e_2, ..., e_m\}$,
G:(V_t, E_t);
while *less than maximum iterations* **do**
  while *v in V* **do**
    if *v not in G* **then**
      add v into G;
      while *e in E that one end is v* **do**
        if *the other end of e is also in G* **then**
          add e to E_t;
        end
      end
    end
    query v based on G;
  end
  if *V is the same as last iteration(converged)* **then**
    break;
  end
end

Algorithm 1: Prediction for MRF

Our classifier based on bag of words has high accuracy, so that the initial assignment of test node was accurate. That makes the algorithm converges fast.

4.4 Results

4.4.1 Overall result

In Table 1, we summarized the results of all methods. From top to down:

1. **Pure Naive Bays**: only utilizing title to classify the topic .
2. **Learning with negative theta**: using citation and title to perform learning without constraining θ to be positive.
3. **Square Loss function**: using citation and title to perform learning and constraining θ to be positive by using Square Loss.
4. **Softmax Loss function**: using citation and title to perform learning and constraining θ to be positive by using Softmax Loss.
5. **Pseudo Likelihood**: using citation and title to perform learning by using Pseudo Likelihood.
6. **Manual Label**: using citation and title but we manually select θ instead of learning. Here, we picked all $\theta^{\oplus} = 1$ and $\theta^{\ominus} = 0.75$ under asymmetric case. And we picked all $\theta = 1$ under symmetric case.

We firstly compared the symmetric and asymmetric relation design. Therefore, we compared methods 2 to 6, which could utilize citations. Except for Pseudo Likelihood, all other methods showed advantage in asymmetric design over symmetric one. Therefore, from this point of view, asymmetric design may bring the extra benefit than symmetric one. Then we found that all methods using citation (no matter symmetric or asymmetric) significantly outperformed the pure Pure Naive Bays method. From this observation, we concluded that utilizing citation relation does bring positive effect in classification. There was a very special case, Pseudo Likelihood, which achieved best performance among all methods, but did not show the advantage of asymmetric design. Therefore,a detail look and analysis are showed in section 4.4.2 for this method.

4.4.2 Results for the model trained in pseudo-likelihood

In the presentation, we only show the result of trained asymmetric R and designed symmetric R and we realized it may cause the difference of performance. Therefore, we averaged the asymmetric parameters as symmetric parameters and tried to test again which shows the performance is almost the same.

The model trained in pseudo-likelihood and gradient ascent way get relatively higher performance in both test set and training set. The training set contains 2491 papers while the test set contains 217 paper.

In the training set, the asymmetric prediction is correct for 2298/2491 papers, and the accuracy is 92.25%. The symmetric prediction is correct for 2293/2491 papers, and the accuracy is 92.05%.

In the test set, the asymmetric prediction is correct for 198/217 papers, and the accuracy is 91.24%. The symmetric prediction is correct for 199/217 papers, and the accuracy is 91.71%.

For R, there is 2 tables in Appendix. For typesetting reason, we don't put it here. There is some interesting fact that we can learn for the asymmetric R. First, papers are more likely to cite other papers but not to be cited by others which makes sense in real life. Second, papers tend to cite and to be cited by the same topic paper. The asymmetric potential shows more power in Neural Network Probabilistic methods and neural network case based. R shows that Neural network tends to cite probabilistic method papers but not to be cited. Case based paper are more likely to cite neural network paper but not to be cited. We can see that in the training set, the asymmetric potential function only gets a little better, and the symmetric potential function even got better performance(1 more correct paper) in test set. There may be several reason for this.

First, the difference between R of cited and citing relation is much smaller than the difference between R of different classes. I think this is the major reason that asymmetric didn't show better performance. Second, there are only several topics show high difference between cited R and citing R and they are not the major of the papers. Third, the title information may do a lot of job. Fourth, because we split the test set by BFS and the test set is a connected component, the distribution of papers may be different from training set. Symmetric model is not very likely to overfit the data, which may also be a reason for this observation.

5. CONCLUSION AND FUTURE WORKS

In this project, we designed a Probabilistic Graphical Model to perform paper classification. We implemented several methods on symmetric and asymmetric design, and performed the comparisons and analyses. Overall, utilizing citation improved the classification result, and the asymmetric design brought the benefit to classification over symmetric design. For the future works, since no fine tune was done in hyper-parameters, we may try to improve our methods by adjusting some hyper-parameters. Moreover, in the view of dataset, we did not use the latest version of CORA and we may try the updated version of CORA or even many other datasets after this course. Finally, we performed our multiple nodes predication inference by using a Gibbs Sampling based method. This method performed very good in this project but we may still try other classical inference methods, like belief propagation, in the future.

6. REFERENCES

[1] R. Crane and L. K. Mcdowell. Investigating markov logic networks for collective classification. In *In ICAART*, 2012.

[2] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[3] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier, 1987.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[5] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, et al. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[6] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[7] L. K. Mcdowell, K. M. Gupta, and D. W. Aha. Case-based collective classification. In *In Proceedings of the Twentieth International FLAIRS Conference. Key West, FL: AAAI*, 2007.

[8] J. Neville and D. Jensen. Iterative classification in relational data. pages 13–20. AAAI Press, 2000.

[9] J. Neville and D. Jensen. Collective classification with relational dependency networks. *Journal of Machine Learning Research*, 8:2007, 2003.

[10] J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 625–630, New York, NY, USA, 2003. ACM.

[11] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1105–1114, New York, NY, USA, 2016. ACM.

[12] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[13] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Table Appendix A1: Citing Parameters learned by pseudo likelihood

R	Cite Case based	Cite Genetic algorithms	Cite Neural networks	Cite Probabilistic methods	Cite Reinforcement learning	Cite Rule learning	Cite Theory
Case based	2.64	0.13	0.37	-0.02	0.54	0.34	0.45
Genetic algorithms	0.12	2.48	-0.04	-0.09	0.48	0.07	-0.46
Neural networks	-0.22	0.34	2.54	1.24	0.18	-0.11	0.53
Probabilistic methods	-0.04	0.01	0.71	2.51	0.24	-0.08	0.66
Reinforcement learning	0.16	0.31	0.12	0.01	2.14	0.10	-0.10
Rule learning	0.46	0.22	-0.32	-0.30	-0.15	2.54	0.53
Theory	0.24	0.03	0.77	0.49	0.69	0.53	1.99

Table Appendix A2: Cited Parameters learned by pseudo likelihood

R	Cited by Case based	Cited by Genetic algorithms	Cited by Neural networks	Cited by Probabilistic methods	Cited by Reinforcement learning	Cited by Rule learning	Cited by Theory
Case based	1.79	0.09	-0.36	0.06	0.42	0.51	0.11
Genetic algorithms	0.31	2.13	0.23	0.04	0.73	0.19	0.37
Neural networks	0.78	-0.02	1.92	0.20	0.33	0.46	0.73
Probabilistic methods	0.18	-0.09	0.96	2.19	-0.02	0.02	0.52
Reinforcement learning	0.22	0.57	0.03	0.08	1.64	0.15	0.56
Rule learning	0.44	-0.18	-0.22	-0.02	0.28	1.8	0.31
Theory	0.29	-0.24	0.66	1.00	-0.04	0.68	1.81