Razvan Kusztos
Girton College
rek43@cam.ac.uk

Computer Science Tripos, Part III

L42: Machine Learning and Algorithms for Data Mining

Title

Report

20 February 2018

Word count: nana

1 Introduction

Recent years have pushed the scientific community towards collecting increasing amounts of data. It is the case of social networks, biological omes (such as the genome, proteome, connectome), ontologies or traffic networks. In the majority of cases, these data are being explored by standard, supervised machine learning techniques. These only manage to capture the data points in isolation. New algorithms have been developed in order to integrate spatial information (e.g. the graph structure).

Often, graph-based datasets consist of a small subset of manually tagged nodes. The goal is to extrapolate and infer the tags for the rest of the nodes. Unlabeled data being more numerous, efficient algorithms require a number of simplifying assumptions. These include the **continuity assumption**, saying that close nodes are more likely to share a label and the **manifold assumption**, saying that the data points lie on a much lower dimensional manifold. [3]

1.1 Problem Description

A way of categorising the learning algorithms in this scenario is whether they are **transductive** or **inductive**. Transductive learning refers to learning solely about information in the training set. On the other hand, inductive learning solves the more general problem of extrapolating to new, unseen points. This current work focuses on the former. The problem can be stated as follows: We are given a graph (V, E) where V is a set of vertices, E is a set of edges. In general, the structure of E is presented in the form of W, a weight matrix (in the current paper, this will only contain 0 and 1 values – the adjacency matrix, notated A). Moreover, each of the vertices(nodes) is represented by a set of features X. Lastly, we consider a partition of the nodes, consisting of the unlabeled nodes V_U and the labeled nodes V_L. Given all these data, we need to find an algorithm that infers the labels for the set V_U.

1.2 Previous Work

Incorporating the information given by of W in order improve the inference accuracy has driven recent research.

A main class of solutions makes direct use of the **continuity assumption**, by choosing to minimise a loss term directly related to the graph structure. This usually takes the form of a graph Laplacian regularisation term.(Zhu et al., 2003 [22]; Zhou et al., 2004 [20]; Belkin et al., 2006 [1]; Weston et al., 2012 [18]). These methods closely resemble the *label propagation* algorithm [21], or the *MAD* [15] algorithm w where the graph structure is the main driver of the inference algorithm.

On the other hand, we find methods that focus on the **manifold assumption**. These aim to find an embedding of the features which reflects edge information. An early example of embedding through a probabilistic generative model can be found in [12]. However, the recent interest in developing new embedding methods was probably brought by the effectiveness of word embeddings in natural language processing tasks [10]. Some methods include *DeepWalk* [13], which forces node embeddings to be close to representations of random walks containing them, *node2vec* [6], which presents a trade-off between breath first search and depthfirst search, or *Planetoid* [19] which combines the regularisation and embedding approaches.

Lastly, the class of algorithms which will be primarily presented for the rest of this paper are the algorithms based on neural networks, specifically on extensions of **convolutional neural networks**. These methods mainly use spectral graph signal processing techniques such as discussed by *Deferrard et al.* [5], *GCN* [9] or the general approach present in *MoNet* [11]. More recently, *Graph Attentional Networks* [17] leverage new research in *self-attention*. In all these works, the graph information is encoded not as part of a loss function or some preprocessing step, but in the neural network itself.

1.3 Next steps

In the remainder of this work, I will present in some detail the seminal paper introducing *GCN* [9]. I will perform experiments that could shed some light on the importance of each of the three main components of such a machine learning algorithm: the features of the nodes, the graph structure and the training labels.

I will lastly suggest and evaluate some potential changes in their pipeline, and discuss possible interpretations.

2 Implementation

2.1 Graph Convolutional Networks [9]

Kipf and Welling [9] have recently introduced a simple, yet effective way of approximating graph convolutions in neural layers. They limit the convolution operation to the *first order neighbourhood*, justified by the need to reduce overfitting. Moreover, they introduce a *renormalisation trick*, avoiding vanishing gradient issues. Altogether, they provide a well behaved and fast to train layer that improves over existing methods by a significant margin.

Their proposed layer has the form:

$$H^{(l+1)} = \sigma\big(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\big) \tag{1}$$

The term $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ turns out to be the normalized Laplacian matrix [7] of the graph with added self-loops. This is what the authors refer to as the *renormalisation trick*. In the above, $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

The paper explains how this model can be derived from the standard graph convolution, by limiting to a local pooling approach. This is put in contrast with the work of *Hammond et al.* [7], where the convolution operation is decomposed into a sum of *Chebyshev polynomials*. The current model is assumed to provide more expressivity (achieved via stacking multiple layers) without being limited by a set parametrisation.

One particular thing to note is that, although the improvement is not huge compared to competing methods such as *Planetoid* [19] or *ICA* [2], it improves by a significant margin over methods that do not take into account any of the graph structure.

However, there are a couple of particularities that the authors suggest as difficulties for this instance of problems. Firstly, their appendix shows how deeply stacking the graph convolutional layer decreases the inference capacities of the model. Secondly, they repeatedly reference overfitting and regularisation as this class of problems is known to suffer from a very small training set. In the case at hand, the training set represents a mere 5% of the whole dataset. As we will see, the performance of their algorithm is dependent on regularisation and *dropout* [14] parameters.

2.2 Technical details

The implementation used throughout the experiments uses Keras [4] and is based on the authors' implementation [1].

I have used the same setup as the original paper [9], with the exception that I changed the activation function (referred to as σ in equation 1) to be the linear activation function. This yielded better results in my experiments.

[1] https://github.com/tkipf/keras-gcn

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \tag{2}$$

$$f(X, A) = softmax\Big(\hat{A}\big(\hat{A}XW^{(0)}\big)W^{(1)}\Big) \tag{3}$$

In order to perform semi-supervised label classification, the categorical crossentropy loss is calculated, where Y_L are the labels of V_L:

$$\mathcal{L} = -\sum_{l \in Y_L}\sum_{f=1}^{F} Y_{lf} ln(f(X, A)_{lf})$$

Finally, the weights $W^{(0)}$ and $W^{(1)}$ are being trained using the Adam optimiser [8].

2.3 Alternative Models

An aim of this mini-paper is to analyse how the different inputs of the problem come together to maximise inference power.

- As a *baseline*, I will use a simple neural network. This entails replacing the graph convolutional layers with dense layers.

  $$f(X) = softmax\big(XW^{(0)}W^{(1)}\big)$$

  As the formula above doesn't encode any information about the graph structure at all, it could be used to put a lower bound on the inference power of the features alone.

  This reduces the problem to the classical supervised learning approach. In general, in fact, it isn't the case that the including the notion of graph structure necessarily increases the predictive capacities. Consider, for example, a graph structure that is trivially derived from the features.

- Next, I will assess a system that combines the training labels and the graph structure alone. For achieving this, I suggest two potential ways. Starting from the *GCN* model, we can replace the input in equation 3 with either the known labels or random data. I expect the convolution operator of Kipf and Welling to achieve results close to techniques such as majority voting or label propagation.

- Since the graph information has not been derived from the feature information (for example by using k-nearest neighbours), intuitively combining them should contain more information than each in individuality. This is addressed by *GCN*.

- An interesting result is that using *dimensionality reduction* by a significant factor doesn't decrease the accuracy of the *GCN*. This is an important step towards reducing the memory requirement of training the model.

3 Evaluation

3.1 Dataset

For the purpose of validating the suggested models, I will be using the Cora dataset, with the same split that was used in Kipf's paper [9].

Table 1: Cora dataset statistics

Dataset	Nodes	Edges	Classes	Features
Cora	2,708	5,429	7	1,433

The Cora dataset consists of scientific papers, each represented using a boolean vector indicating the presence of a vocabulary word in the paper. The graph structure arises from the citation network. For the purpose of these experiments, the structure is symmetrised, so it corresponds to an undirected graph. Lastly, labels correspond to scientific fields the papers belong to.

The training-validation-test split for performing semi-supervised classification task uses the procedure present in Kipf's original implementation [2].

Table 2: Training-test-validation split

Total Nodes	Training	Validation	Test
2,708	140	500	1,000

3.2 Set-Up

I will largely use the same experimental setup as in [9]. I am using the same dataset splits as described in 2. As the system is dependent on regularisation techniques, I will be using *dropout* for all layers, as well as *L2 regularisation* for all trainable weights.

For minimising the loss, I use the Adam [8] optimiser, with a learning rate set to *0.01*. As opposed to Kipf's work, I do not use *early stopping*.

The values reported are the ones maximised using *cross-validation*.

The whole training data is presented to the model in a single batch, and the training is done for 200 epochs.

3.3 Baseline

In the following, I illustrate the results obtained by a vanilla neural network, consisting of three dense layers, the first two with *linear* activations and the last one with a *softmax* activation. I perform a series of preprocessing procedures: 1) unpreprocessed, 2) PCA, 3) neighbour averaging at a distance of one and two edges.

As formulae, the neighbourhood averages are computed as:

$$X_{n.avg.1} = AX$$
$$X_{n.avg.2} = \alpha AX + (1-\alpha)A^2X$$

This analysis shows that simply incorporating information from a close neighbourhood can boost the algorithm's effectiveness by a significant margin. In the above, $\alpha = 0.55$ and the *PCA* reduces the number of features to 50.

3.4 Forgetting the features

In this section, I completely ignore the features and use the *GCN* used in the Kipf's paper [9]. In order to discard the features, I replace the matrix X in two different ways:

[2] https://github.com/tkipf/gcn

Preprocessing	Validation Accuracy	Testing Accuracy
unpreprocessed	55.2	54.6
PCA	52.8	53.9
neighbourhood avg.	60.6	62.6
neighb. avg. 2	76.2	**76.9**
neighb. avg. 2 + PCA	75.8	**74.4**

Table 3: Accuracy results of methods that illustrate importance of the graph structure

Model	Validation Accuracy	Test Accuracy
Random Features [3]	52.4	54.8
Labels as Features	53.8	54.3

Table 4: Accuracy results of models that further prove the information content of the graph structure

- using as input the labels of the training nodes. To match the batch size, I use:

$$y_v = \begin{cases} label(v), & \text{if } v \in V_{training} \\ 0 & \text{otherwise} \end{cases}$$

- feeding in a randomly generated feature matrix with the same shape as X. In order to avoid overfitting, I will run the experiment multiple times and present the average.

These results show that the training labels, together with the adjacency structure alone are capable of discriminating the classes. In fact, this accuracy matches the one obtained by using the features alone.

This provides an intuitive understanding of why the GCN is so effective at solving the problem at hand. It combines the label information extracted from features through the mechanism of the neural network presented in section 3.3, together with the label information extrapolated from the training labels along the graph structure, as in section 3.4.

3.5 GCN experiments and alterations

A first modification that improves the accuracy of the system is simply replacing the nonlinearity described in *Kipf et al.* [9] with a *linear* activation function.

In order to partially address the memory requirement issue, the features can be preprocessed using a *dimensionality reduction* technique. To assess this I will use a variety of PCA procedures.

We can see from Figure 1 that, by keeping 50 principal components, the accuracy of the GCN is not decreased. This is equivalent to a 30-fold decrease in the number of features.

This exercise in dimensionality reduction inspires a more intuitive approach to data exploration. In the following, we can use t-SNE [16] to explore the structure of the learned distribution.

Activation	Validation Accuracy	Test Accuracy
ReLu[4]	79.8	82.2
Tanh	80.4	82.7
SeLu	79.6	81.4
Linear	**80.6**	**83.2**

Table 5: Comparing different activation functions

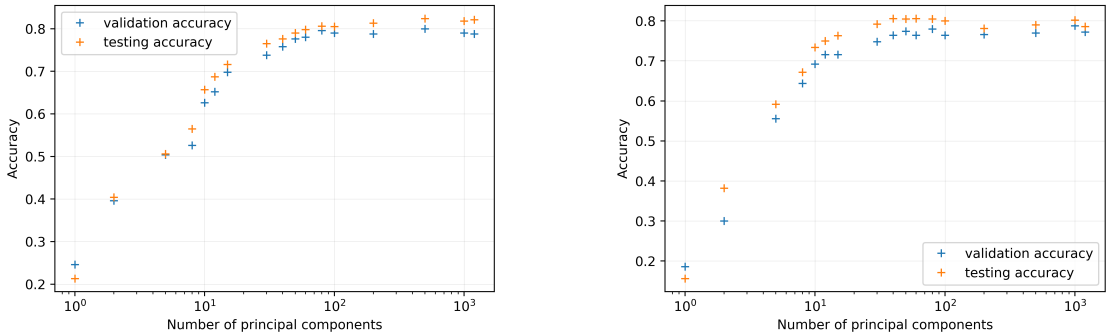

Figure 1: Graphs showing the evolution of accuracy over the number of PCA components used. Left is textbook PCA, Right is Kernel PCA with RBF kernel.

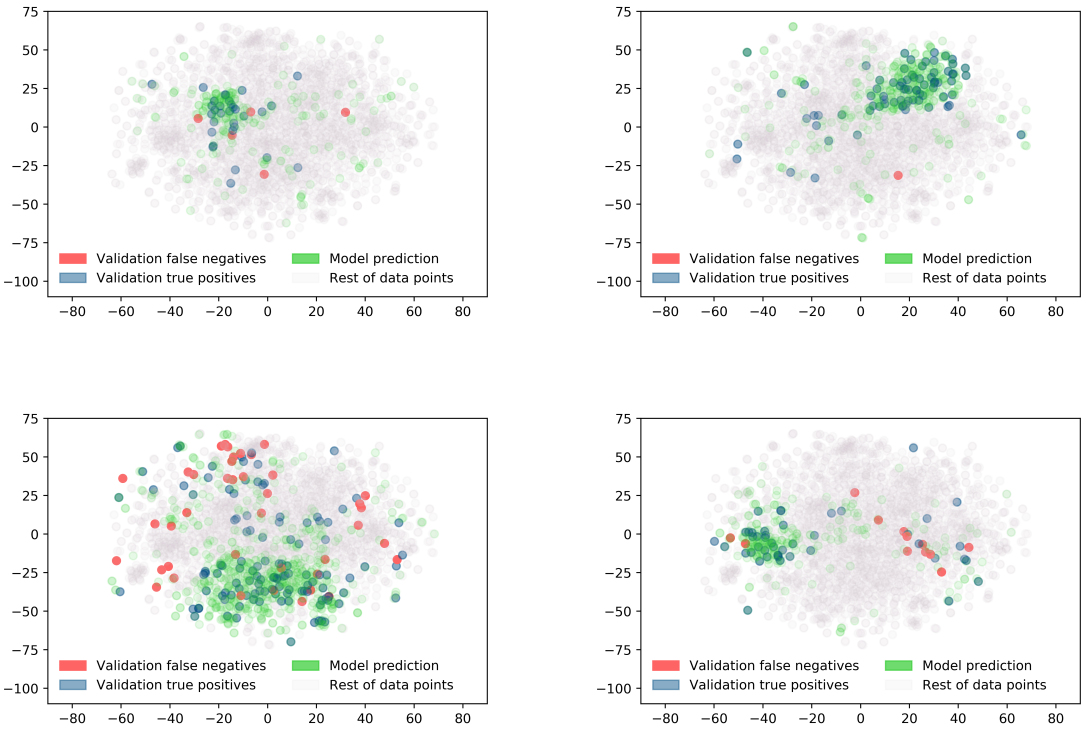

Figure 2: Displaying the prediction for a subset of labels, as well as the validation set corresponding to that label. From left to right, we have labels 1, 2, 3 and 5

From Figure 2, we can draw a few conclusions about the algorithm and the data at hand. Firstly, we can observe that the data present a cluster structure (in the grey data points).

Although with tSNE it is easy to fall prey to pareidolia [5], the cluster structure hypothesis is confirmed by both the experiments in Section 3.3, as well as in the shape of the model prediction for individual labels.

Secondly, by exploring the validation data points, we can see that most of the false negatives are due to points that are far from their clusters. This is most likely because the algorithm does not learn all it can from the graph structure. A reasonable explanation for this lies in the local pooling approximation [9], combined with the noise in the data.

3.6 Discussions

This suggests that the layers should indeed be stacked to reach a higher order neighbourhood. However, this is shown by *Kipf et al.* [9] to be ineffective. The observations above inspired many failed attempts to better integrate the known label data, all of which obtain accuracies marginally lower than the *GCN*:

- Pretraining the network on label data, as in 3.4.
- Using as training labels the output of a majority voting algorithm.
- Training two parallel *GCNs*, concatenating their outputs and feeding that to a third *GCNs*. The two inputs would be features and training nodes. This provides a way to leverage a trade-off between features and label information

However, the apparent limit in accuracy for this classification task (which doesn't go above 83% from what I am aware of) can be justified by the sparsity of training data. In fact, this could be partly because of the way the training set is constructed. Designed to match a random uniform distribution in the whole data set, the degrees of the nodes inherit the power law distribution. This leads to a majority of low degree nodes which have low entropy. However, in practice, this is not necessarily the case, and high degree notes might be more common. For example, human curators are more likely to classify popular publications, creating an informative training dataset.

4 Conclusions

This mini-project has attempted to reconstruct the results of the *GCNs* seminal paper [9] and explore the reasons for its effectiveness.

I have discussed how the problem can be approached from different angles, as well as illustrated lower bounds on the inference power gained from combining the different elements. I analysed the benefits of using dimensionality reduction techniques. Lastly, I considered some shortcomings of the approach, as well as fundamental limits inherent in the problem statement.

[5] https://distill.pub/2016/misread-tsne/

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[3] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[4] FranÃğois Chollet. keras. https://github.com/fchollet/keras, 2015.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[6] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[7] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[11] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017.

[12] Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455):1077–1087, 2001.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[15] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer, 2009.

[16] L Van Der Maaten and G Hinton. Visualizing high-dimensional data using t-sne. journal of machine learning research. *J Mach Learn Res*, 9:26, 2008.

[17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[18] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

[19] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[20] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[21] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

[22] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.