Case Study Report (DATA7703)

Student Name: Yupeng Wu
Student ID: 45960600
Student Email: yupeng.wu@uqconnect.edu.au

Task Description

Whether we are underfitting or overfitting the data is always a key question after the prediction was made during the machine learning process. According to the paper that claims a 'double-descent risk curve' phenomenon, this case study aims to find a similar result using the graph sample and aggregate model (GraphSAGE) for link prediction problem and the random forest model for the classic MNIST problem.

What is a 'double-descent risk curve' phenomenon?

In short, this is a phenomenon that the test risk will have two descents while the choice of H (e.g. the number of nodes in the hidden layer in GraphSAGE) increases from very small to extremely large. If this is true, we are no longer need to take the sweet spot as the only treat. Instead, with the powerful modern computers and advanced models, we are able to dig more information inside the dataset and may achieve a better performance in the prediction.

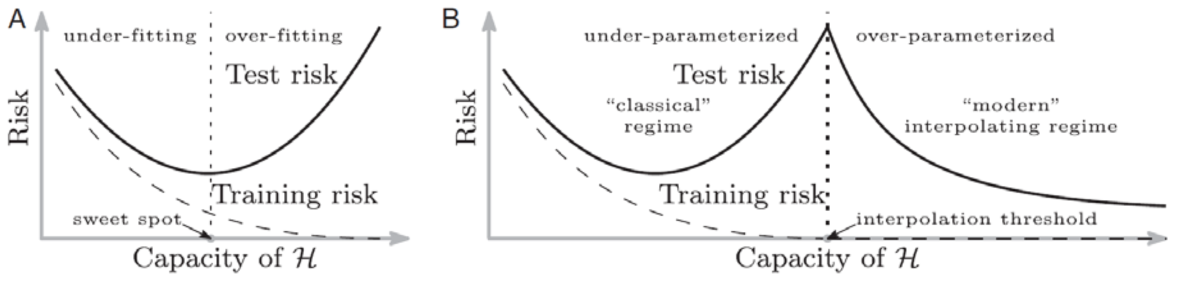


Fig. 1. Curves for training risk (dashed line) and test risk (solid line). (*A*) The classical U-shaped risk curve arising from the bias–variance trade-off. (*B*) The double-descent risk curve, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high-capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

Why using GraphSAGE and random forest model for this case study?

Recently I had another project about using GraphSAGE model for link prediction in network. The paper do not mention whether the GraphSAGE has a 'double-descent risk curve' or not and I would love to find out if the increases of the parameters will help me to achieve a better prediction. Besides the GraphSAGE is a instance of graph convolutional neworks (GCN), which should have a similar performance as a fully connected neural network on MNIST.

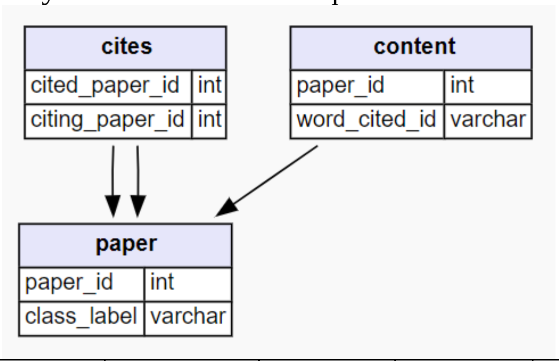
The random forest model, on the other hand, have been experimented in the paper. Since I still doubt the conclusion after reading this paper, it is necessary for me to do the experiment myself.

Although the two models are using different datasets and for different predictions, which means it is hard to do the comparison between their results after the experiment, I am still curioued about the similarity might be happened between the results of those two supervised learning models.

Graph Sample and Aggregate Model

Cora dataset:

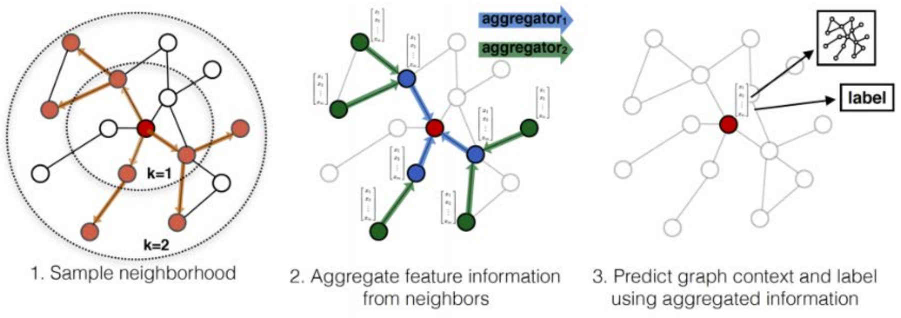
The Cora dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words.



Task	Training	Test	Features	Classes
Classification	976	1084	1433	7

Model introduction:

Instead of using traditional machine learning classification tasks, we can consider using graph neural network (GNN) to perform node classification problems. By providing an explicit link of nodes, this classification problem is no longer classified as an independent problem but leveraging graph structures such as the degree of nodes. The usefulness of graph properties assumes that individual nodes are correlated with other similar nodes.


k=1
k=2
aggregator1
aggregator2
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

GraphSAGE sample and aggregate approach (Hamilton et al., 2018)

Model algorithm:

To address this supervised link prediction problem, build a model with the following architecture using 'stellargraph' in python.

1. Build a two-layer GraphSAGE model that takes labeled node pairs corresponding to possible citation links, and outputs a pair of node embeddings for the citing-paper and cited-paper nodes of the pair.

2. These embeddings are then fed into a link classification layer, which first applies a binary operator to those node embeddings to construct the embedding of the potential link.

3. Thus obtained link embeddings are passed through the dense link classification layer to obtain link predictions - probability for these candidate links to actually exist in the network.

The entire model is trained end-to-end by minimizing the loss function of choice (e.g., binary cross-entropy between predicted link probabilities and true link labels, with true/false citation links having labels 1/0) using stochastic gradient descent (SGD) updates of the model parameters, with minibatches of 'training' links fed into the model.

Random Forest

MNIST dataset:

The MNIST is a large dataset of handwritten digits and the black and white images were normalized to fit into a 28×28 pixel bounding box and antialiased, which introduced grayscale levels.

Task	Training	Test	Features	Classes
Classification	42000	28000	785	10

Model introduction:

Decision trees are a popular method for various machine learning tasks. But those trees that are grown very deep tend to overfit the data with a low bias but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance.

Experiement

Dataset	Training	Test	Features	Classes
Cora	976	1084	1433	7
MNIST	33600	8400	784	10

GraphSAGE model training:

Each model is trained to minimize the loss on the given training set. For the Cora dataset we need to estimate the threshold first. Based on the number of the training data $n = 976$, the number of features $d = 1433$ and the number of the classes $K = 7$.

Based on the formulation in the paper that

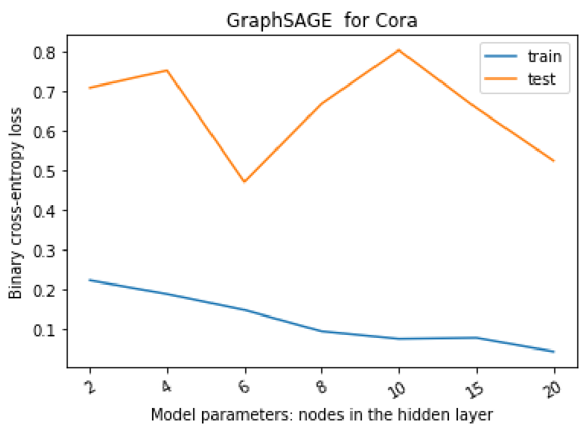
$$(d + 1) \times H + (H + 1) \times K = n \times K$$

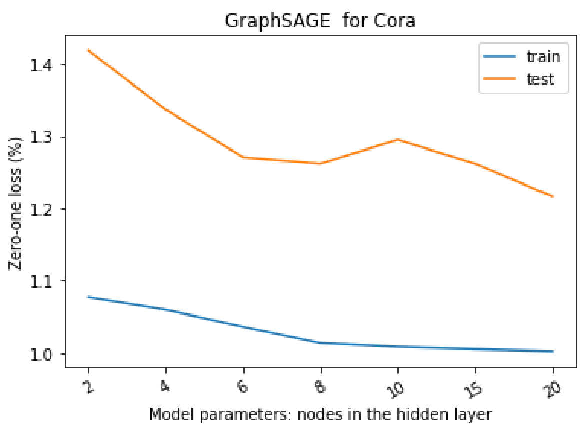
The estimate threshold will be $H \approx 4.8$. We should also consider the fact that the training of the GraphSAGE is very slow (about 13s per epoch) so that we trained the model with 2, 4, 6, 8, 10, 15, 20 nodes in the hidden layer.

Now I found that I made a mistake because the number of the nodes in the hidden layer is smaller than the number of class. And the GraphSAGE is not a fully connected neural network, so I can't use the math form in the paper to estimate the hidden units.

But still I really want to find whether I can perform a 'double-descent risk curve' in Graph SAGE and I continue the experiment.

There is a 2-layer GraphSAGE model acting as node representation learner, with a link classification layer on concatenated node embeddings. Final link classification layer that takes a pair of node embeddings produced by graphsage, applies a binary operator to them to produce the corresponding link embedding ('ip' for inner product), and passes it through a dense layer. Stack the GraphSAGE and prediction layers into a Keras model and specify the loss.





There are two kinds of losses: the binary cross-entropy loss and the zero-one loss. From the results above, it seems that there is a double-descent risk curve for the test risk. But I'm not sure about that because I just did limited experiments and it's hard to say the test risk curve will be always decreasing when the hidden units increasing. So then I set the hidden units range from 5 to 51, trying to show a trend of the binary cross-entropy loss.

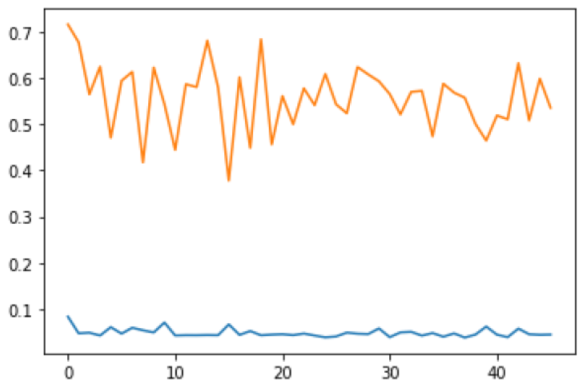


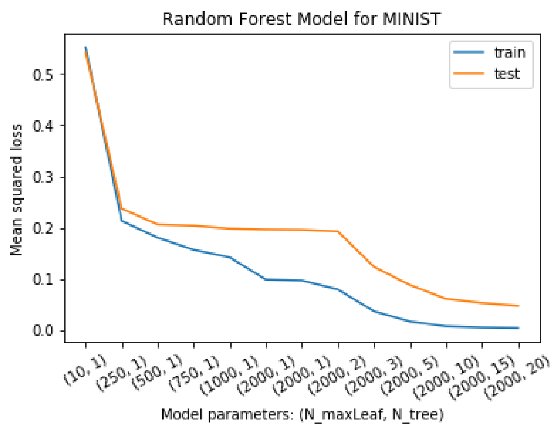
Fig. 2. The trend of the binary cross-entropy loss in GraphSAGE

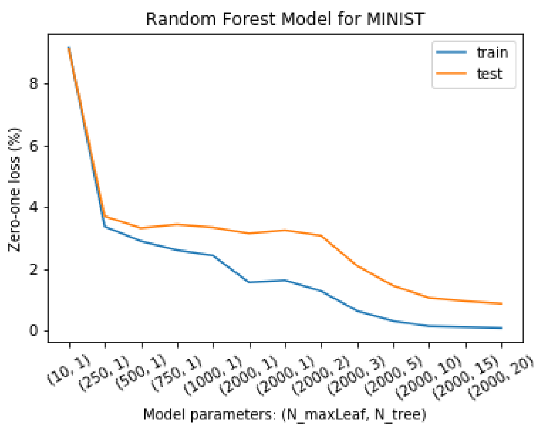
From Fig. 2 we can see that this is a shocked curve. The loss might converage when the parameter keeps increasing, but I'm afraid that this is not a 'double-descent risk curve' phenomenon.

Random forest model training:

For the random forest model the maximum number of leaves and the number of trees are two key parameters which we need to modify. According to the paper, we should only modify the maximum number of leaves and keep the number of trees to be one before the threshold. Theoretically the threshold of the maximum number of leaves will be the number of training data, which in my case, is 33600. But we actually don't need so many leaves in our tree because we might meet the threshold with a smaller number, say 2000 leaves. After we meet the threshold, we should only modify the number of trees and keep the maximum number of leaves fixed.

The squared loss and the zero-one loss are considered for random forest model:





We can see that there is a 'double-descent risk curve' phenomenon in the random forest model. From the above figures we can see that there is a rapid decresing at the beginning (sweet point) and then there is a flat line for test curve. This means we have meet the threshold. And we can see that once we increase the number of trees, another descent happens and the gap between the train loss and the test loss are getting smaller.

Summary and Conclusions

This case study succeeded in proving the 'double-descent risk curve' phenomenon for the random forest model but failed for the graph sample and aggregate model. The result of the random forest is remarkable, but it will be better if I tried more parameters to output the plot.

There are couple reasons for the failure. First, the GraphSAGE is not a fully-connected model so that we can not easily estimate the threshold of the hidden units. Second, there are too many parameters and hyper-parameters for GraphSAGE, including layer size, batch size, etc, so that it is hard to modify them and show a monotonical increase or decrease in the loss with a increasing number of parameters. Besides the GraphSAGE is an advanced and complicated model, so it might be easier for researcher to find a more obvious result with a general multilayer neural network.

Reference

[1]. Reconciling modern machine-learning practice and the classical bias-variance trade-off, Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal

[2]. Random Walk in Node Embeddings (DeepWalk, node2vec, LINE, and GraphSAGE), Edward Ma, https://medium.com/towards-artificial-intelligence/random-walk-in-node-embeddings-deepwalk-node2vec-line-and-graphsage-ca23df60e493

[3]. StellarGraph 1.0.0 documentation

[4]. WIKIPEDIA: MNIST database

[5]. MNIST Classification using Random Forest, Ashwani Jindal, https://www.kaggle.com/ashwani07/mnist-classification-using-random-forest/input?select=train.csv