Autoencoding TPTP

Michael Rawson and Giles Reger

University of Manchester — Manchester, UK
michael@rawsons.uk, regerg@cs.man.ac.uk

Abstract

Extracting features from problem files is a prerequisite in learning systems for automatic theorem proving, notably for strategy creation and scheduling. Such manually-designed features are crucial in enabling machine learning algorithms to help solve otherwise-difficult problems. We propose a neural autoencoder approach for problem sets (allowing automatic feature extraction), and aim to show that the learned features are complementary to human-designed problem features. Learned features may also shed some light on the structure and behaviour of problem sets frequently-used in the community. The TPTP problem set is used as a well-known running example.

1 Background

Given a problem p in a set P, many machine-learning techniques and existing applications require n real-valued features supplied by a feature extraction mapping $f : P \to \mathbb{R}^n$. Learning to predict good prover options ("strategies") is one example of such a system. Previous approaches have often utilised manual feature engineering [2], but this is labour-intensive, and it is not clear in general which features are useful for a given task. Autoencoders [4] learn to reconstruct the input they are given, but must pass data through a "bottleneck" layer which is typically smaller than the input, thereby learning a compressed representation at the bottleneck. Representing the input/output problem set in our application — collections of first-order formulae — is non-trivial, but recent advances in neural network techniques make this more tractable. In this work we use a directed-graph representation of formulae [7], along with graph neural network techniques [1] for encoder and decoder networks.

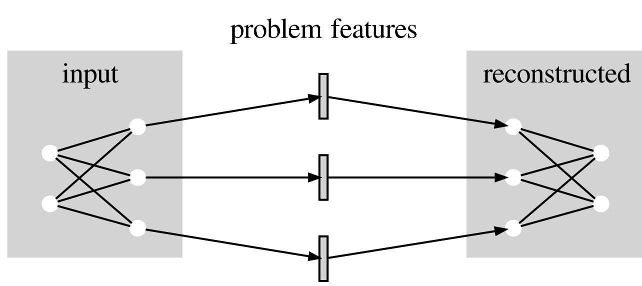

Figure 1: Information flow in the autoencoder. Each problem node receives its own feature vector based on its local formula graph P, and other nodes are then discarded. This vector is then used to try and recover the original information in the reconstructed graph $\hat{P}$.

2 Task

We represent a problem set P as a directed graph. A subset of the nodes in the directed graph are "problem nodes", representing a single problem with constituent axioms and conjecture as immediate children — in TPTP [5] this is a natural construction. An encoder network is allowed to produce a feature vector in $\mathbb{R}^n$ for each node, then all nodes except the problem nodes are discarded in a bottleneck, after which a decoder network attempts to recreate the input graph's node data. A graphical representation of this approach is shown in Figure 1. The level of accuracy in reconstruction and the degree of compression achieved is a useful test of network representation, while also providing a means of producing learned feature vectors from problems in an end-to-end fashion. This transductive task is also interesting from a machine-learning perspective: it is both a node-embedding and autoencoding task. A moderately-deep variational autoencoder model produces reconstruction results significantly better than chance on the first-order problems of TPTP.

3 Future Work

While an obvious next step is to experiment with better neural encoder/decoder pairs, there are many directions for future work. We aim to investigate and present:

1. The effects of different representations and architectures on the performance and on the learned embedding.
2. Conclusions from and visualisations of the learned embedding. Techniques such as t-SNE [3] are expected to be helpful here.
3. Performance of the learned representation on tasks such as strategy scheduling. Are the learned features complementary to existing designed features?
4. Transfer learning: do learned encoders/decoders generalise well to new problem sets? If not, how much training is needed to re-specialise?
5. Comparison of TPTP and other datasets, such as MPTP [6], when viewed under the lens of this new tool.

References

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

[2] Daniel Kühlwein and Josef Urban. MaLeS: A framework for automatic tuning of automated theorem provers. *Journal of Automated Reasoning*, 55(2):91–116, 2015.

[3] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[5] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337, 2009.

[6] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2006.

[7] Mingzhe Wang, Yihe Tang, Jian Wang, and Jia Deng. Premise selection for theorem proving by deep graph embedding. In *Advances in Neural Information Processing Systems*, pages 2786–2796, 2017.