Embeddings Considered Unhelpful?

Lovekesh Vig[1], Ashwin Srinivasan[2], Michael Bain[3], and Ankit Verma[4]

[1] TCS Research, New Delhi
[2] School of Computer Science and Information Systems
BITS Pilani, Goa Campus, Goa 403726
School of Computer Science and Engineering
UNSW, Sydney NSW 2052
School of Computational and Integrative Sciences
Jawaharlal Nehru University, New Delhi

Abstract. Computing similarity in high-dimensional vector spaces is a long-standing problem that has recently seen significant progress with the invention of the *word2vec* algorithm. The basic technique of: (a) using a neural network model to predict words using neighbouring words in sentences from a text corpus; and (b) retrieving vectors of real-numbers (the embeddings) for the words used by the network, has been generalised to obtain vectors for paragraphs, documents, and even non-textual data represented using low-level features. Usually, it has been found that using this embedding results in much better performance for the task being addressed (for example, using documents represented by word-embeddings results in a better classification than representing the documents in the usual bag-of-words manner). It is not known whether embeddings can similarly improve performance with data of the kind considered by Inductive Logic Programming (ILP), in which data apparently dissimilar on the surface, can be similar to each other given domain (background) knowledge. In this paper, for several binary classification tasks, we examine two different approaches to obtaining vector embeddings. In all cases we start with a baseline representation of Boolean features obtained automatically from the data and background knowledge. The first embedding is obtained using sn auto-encoding of the baseline representation, and is class-agnostic. The second is obtained from Siamese networks that maximise distances between the baseline representations classes in the training data (the embedding is thus class-sensitive). We compare classification performance using the baseline representation and the vector embeddings. We find for almost all problems, despite significant efforts to assist the embedding-based models, performance with the baseline features is usually better. We conjecture that this may be because of the use of relevant domain knowledge, which performs the task done by embeddings in other areas, namely, identify semantic similarities. The results obtained here suggest that in such circumstances, there may not be any need to employ vector-space representations.

1 Introduction

The idea of a distance between instances is at the heart of automated methods for classification. This is self-evident for clustering methods that use an explicit distance measure like the Euclidean distance. But even supervised classification methods can often be recast as techniques using more specialised distance calculation. It is therefore not entirely unexpected that with data requiring a high-dimensional representation, the choice of distance measure becomes increasingly important for identifying groups of similar instances. In [1], the authors show that as the data dimension increases, distances can behave unexpectedly. Crucially, for certain obvious measures of distance (the mean-squared distance, for example; and more generally L_k norms for $k > 1$), notions of nearest- and furthest-points become meaningless, and all points are approximately at the same distance from each other. The results concerning high-dimensional data are of relevance to one form of relational learning, in which the data are represented by relational (Boolean) features. In Inductive Logic Programming (ILP), there is a long history (starting from [10]) of addressing classification problems by first identifying Boolean functions of data instances, defined in terms of predicates provided as background knowledge. The values of the functions identified represent a relational instance as a Boolean vector (with the i-th entry for a relational instance x being 1 if the corresponding function $F_i(x) = 1$), augmented with class value of the instance. A straightforward approach would then attempt to build a classification model uaing the class-augmented Boolean vectors as input data. Difficulties can arise, since the Boolean vectors may contain 100's or even 1000's of dimensions, thus bringing up the problems associated with high-dimensional data.

Ignoring for the moment the large body of work on feature-subset selection, at least two kinds of distance-relevant remedies have been proposed to address this issue: (1) In [1], "fractional norms" (L_k norms with

$k < 1$) have been found to be more useful as dimensionality increases; and (2) Distance computations use a representation of data instances as vectors of real-numbers with fewer dimensions. This is similar in spirit to what is done in principal components analysis (PCA), but the new dimensions are the result of non-linear transformations. One way of constructing this non-linear transformation, or embedding, of data instances is via the use of neural networks, most prominently embodied by the *word2vec* algorithm.

Developed by Mikolov *et al* [12], *word2vec* refers to neural network models that are able to embed words into fixed dimensional vector spaces based on their context. The resulting embeddings have been shown to capture some desirable semantic properties with words sharing similar context being mapped to similar vectors. The technique is not restricted to text and embeddings can also be generated in any supervised classification setting in either a class-sensitive or class-agnostic fashion.

In this paper, we investigate the conjecture that neural-net embeddings can assist the classification of relational data instances, represented by Boolean feature-vectors. In our results we do not find evidence to support the conjecture that embeddings improve the classification performance. Specifically, we find neither class-agnostic nor class-sensitive embeddings can perform as well as corresponding models that just use the feature-vector representation of the data. On the face of it, this appears to go against recent positive results on the utility of embeddings for graph-based data (for example, see: [5]). Why is this so? We believe the reason may lie in a key feature of ILP, namely the use of domain knowledge. It is possible that the patterns of semantic similarity achieved by the use of embeddings in other applications is already captured by the domain knowledge predicates provided (and used by the relational features). This suggests a refinement to our (clearly) provocative title, namely: embeddings may not be useful for classification of relational data when there is sufficient amount of relevant domain knowledge.

The rest of the paper is organised as follows. In Section 2.1 we briefly describe how we obtain a first-order Boolean vector-representations of relational data. In Section 2.2 we describe the techniques for obtaining real-vector representations of the relational data. Experimental evaluation of the conjecture is in Section 3. Section 4 concludes the paper.

2 Boolean- and Vector-Space Representations

2.1 First-Order Feature-based Representation

We re-use the approach in [17] for a randomised procedure for obtaining a feature representation. For reasons of space, we do not reproduce the procedure here. Instead we simply describe it's inputs and output. Given: (a) background knowledge B; (b) modes M; (c) relational instances E; (d) language restrictions $\mathcal{L}$; (e) a bound on the depth d of existential variable chains; and (f) a bound on the number of samples N, the procedure returns a sequence of at most N Boolean functions $F_i(x)$ defined in terms of predicates in B. For any given instance $x \in E$, it is assumed that the computation of $F_i(x)$ will terminate and return $TRUE$ or $FALSE$ (we refer the reader to [14] for details of bodels, variable-chain depths and language restrictions). The result is that each relational data instance x is transformed into a fixed size Boolean vector. Clearly, in this representation, each feature maps directly to a first-order statement about the relational instance. The ILP practitioner will recognise this simply as a randomised propositionalisation method (see, for example, [3, 11] for some of the latest developments in this line of research).

2.2 Vector-Space Representation

The objective here is to generate representations of the data as fixed-sized real-valued vectors. We consider two different ways of generating such vectors for classification problems. The first is a class-agnostic approach, in which the representation does not care about the class of the instance; and the second is a class-sensitive approach, which attempts to maximise the distance between vectors of different classes. The class-agnostic representation is generated using an autoencoder. Here a low-dimensional real-vector representation of the data is obtained by using a feedforward neural network that is trained to reconstruct the input data, using one or more intermediate layers with significantly fewer nodes than the input. The class-sensitive embeddings are obtained by learning similarity across samples of the same class using a Siamese network [9]. A Siamese network takes pairs of samples as input and maps each sample into an embedding space via identical base networks. During training the euclidean distance in the embedding space between samples of the same class is minimized and that between samples belonging to different classes is maximized. Networks for the autoencoder and Siamese

embeddings need a feature-based representation to start with: we use the Boolean feature-vector representation of the relational instances as the input for the autoencoder or the Siamese network.

3 Experimental Evaluation

3.1 Aims

We investigate the hypothesis that classification models that use vector-space representations of relational data perform better than those that do not. Specifically, we compare:

– The classification performance obtained using the first-order feature representation against the classification performance obtained using vector-space representations as input where the representations were obtained using class-agnostic and class-sensitive neural-network models.

We will call the first-order feature representation the baseline representation, and the others as embedding representations. By a class-agnostic embedding, we mean an embedding obtained from using an autoencoder of the Boolean feature-vectors constituing the data. By a class-sensitive embedding, we mean an embedding obtained by using a Siamese network trained to separate feature-vectors of positive and negative instances. Classification performance will be assessed using two different kinds of models (a deep neural network, and gradient boosted trees). The models are obtained using values for the baseline and embedding representations as input.

3.2 Materials

Data We report results from experiments conducted using 7 well-studied real world problems from the ILP literature. These are: Mutagenesis [6]; Carcinogenesis [7]; DssTox [13]; and 4 datasets arising from the comparison of Alzheimer's drugs denoted here as *Amine*, *Choline*, *Scop* and *Toxic* [16]. For reasons of space, we refer the reader to the relevant ILP literature with details of the background knowledge available for each problem (this ranges from definitions of generic functional groups, to specific substitutions into template positions for the Alzheimer data sets).

Algorithms and Machines Random features were constructed on an Intel Core i7 laptop computer, using VMware virtual machine running Fedora 13, with an allocation of 2GB for the virtual machine. The Prolog compiler used was Yap. Feature-construction uses the utilities provided by the Aleph ILP system [15] for constructing most-specific clauses in a depth-bounded mode language, and for drawing clauses subsuming such most-specific clauses. No use is made of any of the search procedures within Aleph. The deep networks were constructed using the Keras library with Theano as the backend, and were trained using an NVIDIA K-40 GPU card.

3.3 Method

The main steps of our experimental method are these:

For each dataset
1. Let Tr denote the training set and Te denote the test set
2. Obtain a sequence of features F using Tr
3. Let Tr_F denote the feature-based representation of Tr using values for the F and Te_F be the corresponding feature-based representation of Te (we assume for simplicity that Tr_F and Te_F include the class-values of instances
4. Let M_F be the classification model constructed with Tr_F and A_F be its performance on Te_F
5. Let Tr_V be the vector-space representation of Tr using the embeddings obtained using Tr_F and Te_V be the vector-space representation of Te
6. Let M_V be the classification model constructed with Tr_V and A_V be its performance on Te_V
7. Compare A_F and A_V

The following additional details are relevant:

- For all problems, performance is in fact assessed using 10-fold cross-validation. So the steps above are repeated 10 times for each train-test split.
- For obtaining features, we allow a maximum number of samples of 10,000. This is the same as in [17]. Thus, the maximum dimensionality is bounded by this number.
- We consider two kinds of classification models. First, we use a straightforward deep neural network (DNN). This is a model with multiple, fully connected feedforward layers of rectified linear (ReLU) units followed by Dropout for regularization (see [4] for a description of these ideas). The model weights were initialized with a Gaussian distribution. The number of layers, number of units for each layer, the optimizers, and other training hyperparameters such as learning rate, were determined via a validation set, which is part of the training data. Since the data is limited for the datasets under consideration, after obtaining the model which yields the best validation score, the chosen model is then retrained on the complete training set (this includes the validation set) until the training loss exceeds the training loss obtained for the chosen model during validation. The second kind of model considered are gradient boosted trees as implemented by the XGBoost procedure [2], wherein fixed sized trees are usually the base models, and new tree models are fit on the error residuals at each step, and successively merged with the base model. We used a maximum tree depth of 6, with an L2 regularization paramter of 1, and a step size of 0.3 for our XGBoost model.
- For autoencoders we chose multilayer feedforward neural networks with rectified activation (ReLU) units. A number of hyper-parameters like embedding dimension, learning rate, number of hidden layers, and momentum had to be tuned based on classification performance on a validation set. The number of dimensions were varied from 50 to 500 in steps of 50, the learning rate from 0.001 to 0.01 in steps of 0.001, the Nesterov momentum was kept at 0.9, and the number of hidden layers were either 1 or 3. We used the adam optimizer[8] for training our network. For our Siamese network we chose a deep feedforward network with two hidden layers, and an embedding dimension of 100. The training procedure involved creating 20 training batches comprising of 1000 positive and 1000 negative sample pairs (obtained by sampling with repetition) in addition to 500 positive and 500 negative validation pairs. After every 20 epochs of training, the worst performing class pairs on the validation set were identified and additional pairs from these classes were added to the training set. This was contiued until performance converged on the validation set.
- In all cases, classification performance will be measured by the accuracy obtained from predictions made on the cross-validation test sets.

3.4 Results

The principal results of the empirical evaluation are tabulated in Fig. 1. The main observation that can be made from this tabulation is this: For either kind of classification model, the performance of models using the baseline first-order features are almost always better than models that use an embedding-based representation (either class-agnostic or class-sensitive). This suggests that there is no significant evidence for the utility of embeddings in these experiments.

	Accuracy		
Problem	*Feat*	*Auto*	*Siam*
*Mut*188	0.91(0.06)	0.67 (0.07)	0.91 (0.04)
*Canc*330	0.68(0.03)	0.54 (0.02)	0.60 (0.07)
DssTox	0.70(0.06)	0.62 (0.06)	0.74 (0.05)
Amine	0.89(0.04)	0.63 (0.07)	0.93 (0.03)
Choline	0.81(0.03)	0.50 (0.04)	0.81 (0.05)
Scop	0.80(0.05)	0.51 (0.07)	0.83 (0.08)
Toxic	0.93(0.03)	0.53 (0.06)	0.93 (0.03)

(a) Deep Network

	Accuracy		
Problem	*Feat*	*Auto*	*Siam*
*Mut*188	0.91 (0.06)	76 (0.06)	0.89 (0.05)
*Canc*330	0.64 (0.10)	62 (0.07)	0.61 (0.08)
DssTox	0.75 (0.07)	64 (0.03)	0.73 (0.05)
Amine	0.91 (0.02)	79 (0.03)	0.92 (0.02)
Choline	0.83 (0.03)	58 (0.10)	0.81 (0.04)
Scop	0.91 (0.02)	49 (0.04)	0.80 (0.09)
Toxic	0.93 (0.03)	64 (0.06)	0.93 (0.03)

(b) XGBoost

Fig. 1. Estimated predictive accuracies using the baseline (*Feat*) and embedding (*Auto* and *Siam*)representations. Here, *Feat* with deep networks are identical to the results in [17]. *Auto* denotes embeddings obtained using an autoencoding, and *Siam* denotes embeddings obtained using a Siamese network.

We turn now to the question of why embeddings do not seem to help, despite evidence in other applications of their utility. One point of difference is that presence of domain knowledge for the problems here. This leads us

to conjecture that with sufficient amounts of relevant background knowledge, semantic similarities of the kind thought to be captured by embeddings may already be implicit in the relational features used in the baseline representation. This suggests the following:

Domain-Knowledge Conjecture. Progressively increasing relevant domain-knowledge will make models using an embedding-based representation correspondingly less effective.

We intend to conduct an experiment to test this hypothesis in the future.

4 Concluding Remarks

In this paper we have presented work-in-progress that appears to suggest that there is little to be gained from converting a reasonably straightforward baseline representation of relational data as Boolean vectors into a more compact representation in a space of real-valued vectors. This result, although negative, is still of value since prominent recent successes of embedding-based models like *word2vec* may encourage a practical approach that always prefers an embedding-based representation. What our results suggest is that this is not always so. We believe this may be in some part at least due the extensive use of domain knowledge in the baseline representation.

There are some caveats on the conclusions we have drawn here. Even fixing the background knowledge and the baseline features, the results are still dependent on the classification technique used. Could a different classification technique found better models using the embedding-based representations? We have tried to control for this by using two of the most powerful classification methods available at present (deep neural networks and gradient boosted trees). The results obtained with the baseline representation are the highest known to us at present: so it is likely also that the results with the embedding-based representation are as high as they can be. But what if we used a different embedding method? It is indeed possible that a different embedding technique would have yielded better results. Again, the use of Siamese networks is amongst the most powerful methods available for obtaining embeddings for classification. So, at present, we believe we are using powerful methods for classification and for representation. While this does not guarantee the best overall performance, it does mark out a territory, within which the conclusions drawn are plausible.

Acknowledgements

A.S. is a Visiting Professor in the Department of Computer Science, University of Oxford; and Visiting Professorial Fellow, School of CSE, UNSW Sydney. A.S. is supported by the SERB grant EMR/2016/002766. The authors are grateful to X for running the code for generating results for XGBoost with the Siamese embeddings.

References

1. Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings.*, pages 420–434, 2001.
2. Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.
3. Manoel V. M. França, Gerson Zaverucha, and Artur S. d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.
4. Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.
5. Larry Heck and Hongzhao Huang. Deep learning of knowledge graph embeddings for semantic parsing of twitter dialogs. In *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014, Atlanta, GA, USA, December 3-5, 2014*, pages 597–601, 2014.
6. R. D. King, S. H. Muggleton, A. Srinivasan, and M J Sternberg. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 93(1):438–42, January 1996.
7. R. D. King and A. Srinivasan. Prediction of rodent carcinogenicity bioassays from molecular structure using inductive logic programming. *Environmental Health Perspectives*, 104:pp. 1031–1040, Oct. 1996.
8. Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

9. Gregory Koch. Siamese neural networks for one-shot image recognition. 2015.
10. Nada Lavrac, Saso Dzeroski, and Marko Grobelnik. Learning nonrecursive definitions of relations with LINUS. In *Machine Learning - EWSL-91, European Working Session on Learning, Porto, Portugal, March 6-8, 1991, Proceedings*, pages 265–281, 1991.
11. Huma Lodhi. Deep relational machines. In *Neural Information Processing - 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part II*, pages 212–219, 2013.
12. Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.
13. S. H. Muggleton, J. C. A. Santos, and A. Tamaddoni-Nezhad. Toplog: Ilp using a logic program declarative bias. In M. Garcia de la Banda and E. Pontelli, editors, *Logic Programming*, volume 5366, pages 687–692. 2008.
14. Stephen Muggleton. Inverse entailment and progol. *New Generation Comput.*, 13(3&4):245–286, 1995.
15. A. Srinivasan. The Aleph Manual. Available at http://www.comlab.ox.ac.uk/oucl/ research/areas/machlearn/Aleph/, 1999.
16. A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artif. Intell.*, 85(1-2):277–299, 1996.
17. Ashwin Srinivasan and Lovekesh Vig. Mode-Directed Neural-Symbolic Modelling. Submitted to: The 27th International Conference on Inductive Logic Programming (ILP2017), 2017.