Ontology Graph Embeddings and ILP for Financial Forecasting

30th International Conference on Inductive Logic Programming

Can Erten and Dimitar Kazakov
AI Group, CS, University of York, UK

Athens, Greece
25-27 October 2021

Abstract

There is a history of hybrid machine learning approaches where the result of an unsupervised learning algorithm is used to provide data annotation from which ILP can learn in the usual supervised manner. Here we consider the task of predicting the Boolean property of cointegration between the time series of stock price of two companies, which can be used to implement a robust pair-trading strategy that can remain profitable regardless of the overall direction in which the market evolves.

We start with an original FinTech ontology of relations between companies and their managers, which we have previously extracted from SEC reports, quarterly filings that are mandatory for all US companies. When combined with stock price time series, these relations have been shown to help find pairs of companies suitable to pair trading. Here we use node2vec embeddings to produce clusters of companies and managers, which are then used as background predicates. Background knowledge also includes the relations linking companies and staff present in the ontology. Progol is used to learn from this mixture of predicates combining numerical with structural relations between the entities represented in the data set to reveal rules with predictive power.

SEC Reports

- U.S. Securities and Exchange Commission (SEC): an independent agency for the federal government
- Requires a mandatory report from all U.S. companies
- Several forms on the company's activities and trades:
  - Company statements: annual, quarterly and monthly;
  - Reports of a change of structure, etc.
- Reports used here: on "corporate insiders," employees buying or selling more than 10% of the company's shares (due within 10 days of transaction)
- Represented as strongly typed XML with the schema. Convenient index available on the SEC server (for company name, period, type of report).
- We have generated knowledge graphs from this data (Forms 3-5) for each 3-month period to represent links between companies and people.

UNIVERSITY of York

Ontology

- A structured database representing objects (aka *individuals*) and their relations (aka *roles*)
- Base unit: a triplet <*subject, predicate, object*>
- Individuals are grouped into classes (aka *concepts*) with a hierarchy. Role hierarchy also possible.
- Formal representations of knowledge with well-defined semantics based on Description Logics (DLs).
- ILP learners using DL for data and models exist.

Node2Vec Embeddings

- Graph Representation Learning: apply existing machine learning algorithms, such as CNN and RNN on graph data.
- The graph need to be translated to the vector space, and represented as a tensor.
- Node2vec generates real-valued vector representations (embeddings) of nodes on a graph. Distance then reflects a mixture of neighbourhood proximity and local topology similarity.

Clustering Ontology Nodes

- We use node2vec embeddings of the SEC ontology nodes to cluster them with k-means.
- Then each node can be assigned to one cluster in unsupervised way.
- A predicate is defined to assign each node its the cluster number and is then used as background knowledge to an ILP (supervised) learner: Progol.

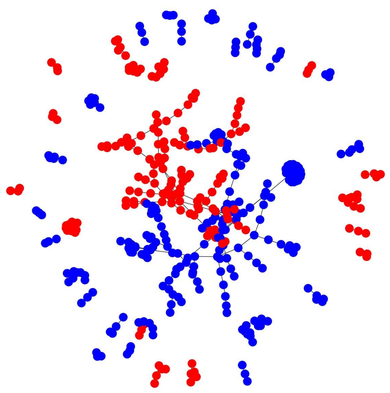

Figure: Nodes of 2 clusters and their ontology connections.

Experiment Design

Prolog/Progol source files: automatically generated for each experiment. The following steps are taken:

- Load the serialised ontology data
- Run a query to select companies (background information);
- Run a query to select people linked with the companies (background information);
- Split the data into a training set and a test set
- Run the cointegration test for all pairs of companies to generate positive and negative examples
- Run the node2vec algorithm on the data to generate node embeddings;
- Run k-means on the embeddings to obtain clusters (to be used as background knowledge)
- Generate a Progol source file from the background knowledge and target predicate examples
- Run Progol on the data

Table 1. Dataset description

#of Training Companies	700
#of Test Companies	300
Positive Training Pairs	8,833
Negative Training Pairs	235,817
Positive Test Pairs	1,405
Negative Test Pairs	43,445

- A total of 1948 companies for the Q3/2017 period.
- 1000 companies were selected and split 70% : 30% at random.
- All 244,650 possible pairs formed by companies in the first set were earmarked as training data,
- 44,850 pairs formed by companies in the remaining set were used as test data.
- We applied a cointegration test to each pair of companies
- But opted for positive-only learning for reasons of both speed and accuracy.
- Percentage of cointegrated pairs in test data: 6.1%

```
:- modeh(1,coint(+company, +company))?
:- modeb(1,ccluster(+company,#int))?
:- modeb(1,ccon2(+company,#int))?
:- modeb(1,pcon2(+person,#int))?
:- modeb(1,connect(+company,+company))?
:- modeb(1,connect(+company,+company))?
```

Results

Rules	pos	neg	$\frac{pos}{pos+neg}$	p-value
coint(A,B) :- ccluster(A,10), ccluster(B,13).	5	44	0.10	**.0046**
coint(A,B) :- ccluster(A,5), ccluster(B,15).	9	43	0.17	**.0000**
coint(A,B) :- ccluster(A,5), ccluster(B,10).	6	46	0.12	**.0005**
coint(A,B) :- ccluster(A,5), ccluster(B,9).	6	53	0.10	**.0020**
coint(A,B) :- ccluster(A,5), ccluster(B,11).	9	52	0.15	**.0000**
coint(A,B) :- ccluster(A,9), ccluster(B,11).	7	57	0.11	**.0003**
coint(A,B) :- ccluster(A,9), ccluster(B,15).	3	58	0.05	.4239
coint(A,B) :- ccluster(A,9), ccluster(B,10).	2	53	0.04	.8303
coint(A,B) :- ccluster(A,8), ccluster(B,15).	4	34	0.11	**.0090**
coint(A,B) :- ccluster(A,14), ccluster(B,15).	4	84	0.05	.4474
coint(A,B) :- ccluster(A,11), ccluster(B,15).	10	101	0.09	**.0004**
coint(A,B) :- ccluster(A,10), ccluster(B,11).	6	45	0.12	**.0004**
coint(A,B) :- ccluster(A,5), ccluster(B,8).	4	56	0.07	.1166
coint(A,B) :- ccluster(A,10), ccluster(B,10).	1	35	0.03	.9027
coint(A,B) :- ccluster(A,10), ccluster(B,15).	5	41	0.11	**.0026**
coint(A,B) :- ccluster(A,6), ccluster(B,15).	7	93	0.07	**.0268**
coint(A,B) :- ccluster(B,15), ccon2(A,4).	40	507	0.07	**.0000**
coint(A,B) :- ccluster(A,6), ccon2(A,2), ccon2(B,4).	26	694	0.04	.4652
coint(A,B) :- ccluster(A,7), ccluster(B,7), ccon2(B,3).	8	74	0.10	**.0006**
coint(A,B) :- ccluster(A,6), ccluster(B,6), ccon2(A,4).	0	60	0.00	.1636
coint(A,B) :- ccluster(B,13), ccon2(A,310).	4	25	0.14	**.0010**
coint(A,B) :- ccluster(A,10), ccon2(B,310).	3	7	0.30	**.0000**
coint(A,B) :- ccluster(A,13), ccluster(B,13), ccon2(A,5).	4	62	0.06	.1727
coint(A,B) :- ccluster(A,5), ccluster(B,5), ccon2(B,4).	0	34	0.00	.2944
coint(A,B) :- ccluster(A,9), ccluster(B,9), ccon2(B,2).	0	40	0.00	.2554
coint(A,B) :- ccluster(A,15), ccluster(B,15), ccon2(B,2).	0	37	0.00	.2740

Q3/2017			
	A	¬A	Total
P	1,361	22,192	23,553
¬P	44	21,253	21,297
Total	1,405	43,445	44,850
Accuracy 50.42% ± 0.24%			
χ^2 probability 0.0000			

Conclusion

- Resulting method is a hybrid learning approach:
  = subsymbolic+symbolic & unsupervised+supervised.
- Resulting rules:
  = interpretable relations based on numerical concepts.
- Predictive power:
  = doubles the percentage of cointegrated pairs in test data examples. Almost all ⊕ pairs are retained; 50% of ⊖ pairs rejected - saving the need for costly `coint` test.

Future Work

- Add relations extracted from free text rather than tabular data as at present
- Use a concurrent learner