Learning Representations of Relational Data

Sebastijan Dumančić

DTAI, CS Department, KU Leuven

September 6, ILP 2017
1 Introduction

2 Where are we now?

3 What can we do better?

4 Similarity of relational objects

5 Experiments and results

6 Auto-encoding logic programs
A Few Useful Things to Know about Machine Learning

Pedro Domingos
Department of Computer Science and Engineering

the probability that a bad classifier is consistent with $n$ random, independent training examples is less than $(1 - \epsilon)^n$. Let $b$ be the number of bad classifiers in the learner’s hypothesis space $H$. The probability that at least one of them is consistent is less than $b(1 - \epsilon)^n$, by the union bound. Assuming the learner always returns a consistent classifier, the probability that this classifier is bad is then less than $|H|(1 - \epsilon)^n$, where we have used the fact that $b \leq |H|$. So if we want this probability to be less than $\delta$, it suffices to make $n > \ln(\delta/|H|)/\ln(1 - \epsilon) \geq \frac{1}{\epsilon}(\ln |H| + \ln \frac{1}{\delta})$.

Unfortunately, guarantees of this type have to be taken with a large grain of salt. This is because the bounds obtained in the probabilistic analysis are often loose. A more accurate estimate of the probability of failure is needed, and this is precisely what we will focus on in this tutorial.

\textit{Caveat emptor:} learning is a complex phenomenon, and just because a learner has a theoretical justification and works in practice doesn’t mean the former is the reason for the latter.

8. \textbf{FEATURE ENGINEERING IS THE KEY}
At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. If you have many independent features that each correlate well with the class, learning is easy. On the other hand, if the class is a very complex function of the features, you may not be able to learn it. Often, the raw data is not in a form that is
Deep learning - finding good features autonomously by gradually building complexity
1 – Focus on sensory data

Learning relational latent features – Dumančić, Blockeel
What about relational data?
1 – Relational deep learning?

Developer Survey Results 2017

- MySQL: 55.6%
- SQL Server: 38.6%
- SQLite: 26.6%
- PostgreSQL: 26.5%
- MongoDB: 21.0%
- Oracle: 16.5%
- Redis: 14.1%
- Cassandra: 3.1%
1. Introduction
2. Where are we now?
3. What can we do better?
4. Similarity of relational objects
5. Experiments and results
6. Auto-encoding logic programs
Replace symbols with numbers, and logic with algebra

Entities:
- barack
- michele
- honolulu

Relations:
- wasBornIn(., .)
- isMarriedTo(., .)
- locatedIn(., .)
Learning representation = learning vectors

[ . . . . . . . ]

Entities:
- barack
- michele
- honolulu

Relations:
- wasBornIn(., ., .)
- isMarriedTo(., ., .)
- locatedIn(., ., .)
wasBornIn(barack, honolulu).

\[
\begin{bmatrix}
  \text{barack} \\
  \text{wasBornIn} \\
  \text{honolulu}
\end{bmatrix}^T \approx 1
\]

wasBornIn(barack, nairobi).

\[
\begin{bmatrix}
  \text{barack} \\
  \text{wasBornIn} \\
  \text{nairobi}
\end{bmatrix}^T \approx 0
\]
efficient

good performance on KB completion tasks

uninterpretable latent spaces

huge amounts of data

problems with unseen entities

does not integrate in (statistical) relational learning
1 Introduction

2 Where are we now?

3 What can we do better?

4 Similarity of relational objects

5 Experiments and results

6 Auto-encoding logic programs
3 – Desirable features

Relational  Interpretable

\texttt{implies(\text{logic,interpretability})}.

Symbolic  Unsupervised
Learning relational latent features – Dumančić, Blockeel
Questions:
What to cluster? How to cluster? Architecture?
What to cluster?

For each type/domain of vertices in data

cluster vertices and relationships!
How to cluster them?

Unsupervised approach - which similarity is useful? (features, proximity, struc,...)

Cluster with a diverse set of similarities

Learning relational latent features – Dumančić, Blockeel
How to choose the architecture?

Rely on clustering selection to choose a good clustering.

a predicate for each latent feature
1. Introduction
2. Where are we now?
3. What can we do better?
4. Similarity of relational objects
5. Experiments and results
6. Auto-encoding logic programs
How similar are ProfA and ProfB?

Relational clustering over neighbourhood trees [Dumančić & Blockeel, MLJ 2017]
Neighbourhood trees summarize the neighbourhood of an instance/example data.
Neighbourhood trees summarize the neighbourhood of an instance/example.

Data:
- Level
- TaughtBy
- AdvisedBy
- HasPosition
- InPhase

Neighbourhood tree:
- Level
- TaughtBy
- AdvisedBy
- HasPosition
- InPhase

Similarity of instances = similarity of their neighbourhood trees.

Learning relational latent features – Dumančić, Blockeel
Decompose neighbourhood trees into semantic parts

Graphical representation:
- Nodes: ProfA, StudA, StudB, ProfB, CourseA
- Edges: AdvisedBy, TaughtBy, InPhase, Level, HasPosition
- Root features: ProfA, StudA, StudB
- Features of neighbouring instances: CourseA, Level
- Neighbour identities: StudA, StudB, ProfB
- Edge types: AdvisedBy, TaughtBy, InPhase
- Proximity: Level, HasPosition

Equation:
\[ \text{similarity} = \text{linear combination of similarities of individual semantic parts} \]
Decompose neighbourhood trees into semantic parts

similarity = linear combination of similarities of individual semantic parts
Decompose NT is multisets of:
- attribute
- edge labels
- vertex identities

per level and vertex type

Multiset of edge labels (level 1):
{ (Advised,2), (Advised,2), (TaughtBy,2) }

Compare two multisets, $A$ and $B$ with $\chi^2$ distance

$$\chi^2(A, B) = \sum_{x \in A \cup B} \frac{(f_A(x) - f_B(x))^2}{f_A(x) + f_B(x)}$$
(Hyper)edge similarity – reduction to similarities of vertices

1. Merging

\[
\begin{align*}
\left( \begin{array}{c}
\text{Node 1} \\
\text{Node 2}
\end{array} \right) & \quad \rightarrow \quad \begin{array}{c}
\text{Merged Node}
\end{array} \\
\left( \begin{array}{c}
\text{Node 3} \\
\text{Node 4}
\end{array} \right) & \quad \rightarrow \quad \begin{array}{c}
\text{Merged Node}
\end{array}
\end{align*}
\]

\[
\text{MERGE} \rightarrow \text{SIM} \left( \begin{array}{c}
\text{Merged Node} \\
\text{Node 1, Node 2, Node 3, Node 4}
\end{array} \right)
\]

2. Combination

\[
\begin{align*}
\left( \begin{array}{c}
\text{Node 1} \\
\text{Node 2}
\end{array} \right) & \quad = \quad \text{SIM} \left( \begin{array}{c}
\text{Node 1, Node 2}
\end{array} \right) \\
\left( \begin{array}{c}
\text{Node 3} \\
\text{Node 4}
\end{array} \right) & \quad = \quad \text{SIM} \left( \begin{array}{c}
\text{Node 3, Node 4}
\end{array} \right)
\end{align*}
\]

\[
\text{SIM} \left( \begin{array}{c}
\text{Node 1, Node 2}
\end{array} \right) + \text{SIM} \left( \begin{array}{c}
\text{Node 3, Node 4}
\end{array} \right)
\]

Learning relational latent features – Dumančić, Blockeel
1. Introduction

2. Where are we now?

3. What can we do better?

4. Similarity of relational objects

5. Experiments and results

6. Auto-encoding logic programs
Datasets:
- IMDB
- UWCSE
- Mutagenesis
- Hepatitis
- Terrorist attacks
- WebKB

Setup:
- 5-fold cross validation
- learning features of train data
- mapping test data to the obtained clusters
- learn TILDE models on latent/original representations

Question:
Does learning in relational latent spaces benefits leaning compared to learning in the original space?

- lower model complexity
- increased performance

How does it compared to MRC [Kok & Domingos, ICML 07]
Models learned on latent representations are substantially simpler.

Models learned on latent representations often perform better, except: relationship info not useful.
1. Introduction
2. Where are we now?
3. What can we do better?
4. Similarity of relational objects
5. Experiments and results
6. Auto-encoding logic programs
Logic programs a computational framework for encoder and decoder

**Input:**
- mother(anna, dirk).
- female(anna).
- father(tom, dirk).
- male(tom).

**Encoder:**
- latent1(X, Y) :- mother(X, Y); father(X, Y).
- latent2(X) :- female(X).

**Latent rep.:**
- latent1(anna, dirk).
- latent1(tom, dirk).
- latent2(anna).

**Decoder:**
- mother(X, Y) :- latent1(X, Y), latent2(X).
- female(X) :- latent2(X).
- father(X, Y) :- latent1(X, Y), not(latent2(X)).
- male(X) :- not(latent2(X)).

**Output:**
- mother(anna, dirk).
- female(anna).
- father(tom, dirk).
- male(tom).