End-to-End Differentiable Proving

Tim Rocktäschel

Whiteson Research Lab, University of Oxford

http://rockt.github.com  Twitter: @_rockt  tim.rocktaschel@cs.ox.ac.uk

Logic and Learning Workshop at The Alan Turing Institute

January 12, 2018
What vegetable is on the plate?
Neural Net: broccoli
Ground Truth: broccoli

What color are the shoes on the person's feet?
Neural Net: brown
Ground Truth: brown

How many school busses are there?
Neural Net: 2
Ground Truth: 2

What sport is this?
Neural Net: baseball
Ground Truth: baseball

What is on top of the refrigerator?
Neural Net: magnets
Ground Truth: cereal

What uniform is she wearing?
Neural Net: shorts
Ground Truth: girl scout

What is the table number?
Neural Net: 4
Ground Truth: 40

What are people sitting under in the back?
Neural Net: bench
Ground Truth: tent
a) Chemical Representation of the Synthesis Plan

Target:

\[
\text{Boc} \quad \text{N} \quad \text{O} \quad \text{CO}_2\text{Me}
\]

\[
\begin{align*}
\text{MeO}_2\text{C} \quad \text{CO}_2\text{Me} \\
\text{Ph} \quad \text{OH} \\
\text{2} \\
\text{Boc} \quad \text{N} \quad \text{OH} \\
\text{Boc} \quad \text{N} \quad \text{OTBS} \\
\text{3} \\
\text{Boc} \quad \text{N} \quad \text{OTBS} \\
\text{4} \\
\text{Ph} \quad \text{Br} \\
\text{5} \\
\end{align*}
\]

\[
\begin{align*}
\text{Boc}_2\text{O} + \text{HN} \quad \text{O} \quad \text{CO}_2\text{Me} \\
\text{8} \\
\text{HN} \quad \text{OH} + \text{Boc}_2\text{O} \\
\text{7} \\
\end{align*}
\]

b) Search Tree Representation

Root (Target):

- A
  - B
    - C
      - D

Terminal solved state:

- A = \{1\}
- B = \{2,6\}
- C = \{3,6\}
- D = \{4,5,6\}
- E = \{8,9\}
- F = \{7,8\}

Mit der Maßnahme soll sichergestellt werden, dass die Polizei die lebensrettende Ausrüstung bekommt, die sie braucht, um ihren Job zu machen, sagte US-Justizminister Jeff Sessions.

The police in the USA are allowed to get heavy equipment and weapons from the military again. This was decided by US President Donald Trump, who overturned an order from his predecessor Barack Obama, according to which the Department of Defense was banned from equipping the police with grenade launchers, armoured vehicles, bayonets, large-calibre weapons and ammunition.

The measure is designed to ensure that the police get the lifesaving equipment they need to do their job, US Attorney General Jeff Sessions said.
Die Polizei in den USA darf sich die Waffen beim Militär besorgen. Trump entschieden und so ein Barack Obama aufgehoben, man Verteidigungsministerium verkaufen, Lasergranatwerfer, gepanzerten Fahrzeugen, Bajonetten, großkalibren Waffen und Munition auszurüsten.

Mit der Maßnahme soll sichergestellt werden, dass die Polizei die lebensrettende Ausrüstung bekommt, die sie brauch, um ihren Job zu machen, sagte US-Justizminister Jeff Sessions.

The measure is designed to ensure that the police get the lifesaving equipment they need to do their job, US Attorney General Jeff Sessions said.
THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

XKCD, 17th May 2017
**Data & Explanations**
- Rules
- (Partial) Programs
- Natural Language

---

**XKCD, 17th May 2017**

*This is your machine learning system?*

YUP! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

What if the answers are wrong?

Just stir the pile until they start looking right.
XKCD, 17th May 2017

Data & Explanations
- Rules
- (Partial) Programs
- Natural Language

Answers & Explanations
- Rules
- Programs
- Natural Language
- Plans
THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.

XKCD, 17th May 2017
Expert Systems

• No/little training data
• Interpretable
• Rules manually defined
• No generalization

Neural Networks

• Trained end-to-end
• Strong generalization
• Need a lot of training data
• Not interpretable
goal problem.

rule 1
  if not turn_over and
  battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
  if turn_over and
  smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
  gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
  gas_gauge is low
  then problem is out_of_gas cf 30.
Expert Systems

- No/little training data
- Interpretable

---

rule 1
if not turn_over and
battery_bad
then problem is battery cf 100.

rule 2
if lights_weak
then battery_bad cf 50.

rule 3
if radio_weak
then battery_bad cf 50.

rule 4
if turn_over and
smell_gas
then problem is flooded cf 80.

rule 5
if turn_over and
gas_gauge is empty
then problem is out_of_gas cf 90.

rule 6
if turn_over and
gas_gauge is low
then problem is out_of_gas cf 30.
Expert Systems

- No/little training data
- Interpretable
- Rules manually defined
- No generalization

Prolog and Natural-Language Analysis

Fernando C.N. Pereira and Stuart M. Shieber
goal problem.

rule 1
  if not turn_over and
  battery_bad
  then problem is battery cf 100.

rule 2
  if lights_weak
  then battery_bad cf 50.

rule 3
  if radio_weak
  then battery_bad cf 50.

rule 4
  if turn_over and
  smell_gas
  then problem is flooded cf 80.

rule 5
  if turn_over and
  gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
  if turn_over and
  gas_gauge is low
  then problem is out_of_gas cf 30.

---

**Expert Systems**

- No/little training data
- Interpretable
- Rules manually defined
- No generalization
Expert Systems
- No/little training data
- Interpretable
- Rules manually defined
- No generalization

Neural Networks
- Trained end-to-end
- Strong generalization

goal problem.

rule 1
if not turn_over and
    battery_bad
then problem is battery cf 100.

rule 2
if lights_weak
    then battery_bad cf 50.

rule 3
if radio_weak
    then battery_bad cf 50.

rule 4
if turn_over and
    smell_gas
then problem is flooded cf 80.

rule 5
if turn_over and
    gas_gauge is empty
then problem is out_of_gas cf 90.

rule 6
if turn_over and
    gas_gauge is low
then problem is out_of_gas cf 30.
Expert Systems

• No/little training data
• Interpretable
• Rules manually defined
• No generalization

Neural Networks

• Need a lot of training data
• Not interpretable
• Trained end-to-end
• Strong generalization

rule 1
if not turn_over and
  battery_bad
then problem is battery cf 100.

rule 2
if lights_weak
then battery_bad cf 50.

rule 3
if radio_weak
then battery_bad cf 50.

rule 4
if turn_over and
  smell_gas
then problem is flooded cf 80.

rule 5
if turn_over and
  gas_gauge is empty
then problem is out_of_gas cf 90.

rule 6
if turn_over and
  gas_gauge is low
then problem is out_of_gas cf 30.
Expert Systems

- No/little training data
- Interpretable

Neural Networks

- Trained end-to-end
- Strong generalization

rule 1
if not turn_over and
  battery_bad
then problem is battery cf 100.

rule 2
if lights_weak
  then battery_bad cf 50.

rule 3
if radio_weak
  then battery_bad cf 50.

rule 4
if turn_over and
  smell_gas
  then problem is flooded cf 80.

rule 5
if turn_over and
  gas_gauge is empty
  then problem is out_of_gas cf 90.

rule 6
if turn_over and
  gas_gauge is low
  then problem is out_of_gas cf 30.
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007)...

- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)

- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (d’Avila Garcez and Zaverucha, 1999)
  - First-order inference (no training of symbol representations): Unification Neural Networks (Hölldobler, 1990; Komendantskaya, 2011), SHRUTI (Shastri, 1992), Neural Prolog (Ding, 1995), CLIP++ (Franca et al., 2014), Lifted Relational Networks (Sourek et al., 2015)
  - Recent: Logic Tensor Networks (Serafini and d’Avila Garcez, 2016), TensorLog (Cohen, 2016), Differentiable Inductive Logic (Evans and Grefenstette, 2017)

For overviews see Besold et al. (2017) and d’Avila Garcez et al. (2012)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
- Neural-symbolic Connectionism

For overviews see Besold et al. (2017) and d'Avila Garcez et al. (2012)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (d’Avila Garcez and Zaverucha, 1999)
Machine Learning & Logic

- **Fuzzy Logic** (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) ...
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (d’Avila Garcez and Zaverucha, 1999)
  - First-order inference (no training of symbol representations): Unification Neural Networks (Hölldobler, 1990; Komendantskaya, 2011), SHRUTI (Shastri, 1992), Neural Prolog (Ding, 1995), CLIP++ (Franca et al., 2014), Lifted Relational Networks (Sourek et al., 2015)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (d’Avila Garcez and Zaverucha, 1999)
  - First-order inference (no training of symbol representations): Unification Neural Networks (Hölldobler, 1990; Komendantskaya, 2011), SHRUTI (Shastri, 1992), Neural Prolog (Ding, 1995), CLIP++ (Franca et al., 2014), Lifted Relational Networks (Sourek et al., 2015)
- Recent: Logic Tensor Networks (Serafini and d’Avila Garcez, 2016), TensorLog (Cohen, 2016), Differentiable Inductive Logic (Evans and Grefenstette, 2017)
Machine Learning & Logic

- Fuzzy Logic (Zadeh, 1965)
- Probabilistic Logic Programming, e.g.,
  - IBAL (Pfeffer, 2001), BLOG (Milch et al., 2005), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007) . . .
- Inductive Logic Programming, e.g.,
  - Statistical Predicate Invention (Kok and Domingos, 2007)
- Neural-symbolic Connectionism
  - Propositional rules: EBL-ANN (Shavlik and Towell, 1989), KBANN (Towell and Shavlik, 1994), C-LIP (d’Avila Garcez and Zaverucha, 1999)
  - First-order inference (no training of symbol representations): Unification Neural Networks (Hölldobler, 1990; Komendantskaya, 2011), SHRUTI (Shastri, 1992), Neural Prolog (Ding, 1995), CLIP++ (Franca et al., 2014), Lifted Relational Networks (Sourek et al., 2015)
- Recent: Logic Tensor Networks (Serafini and d’Avila Garcez, 2016), TensorLog (Cohen, 2016), Differentiable Inductive Logic (Evans and Grefenstette, 2017)

For overviews see Besold et al. (2017) and d’Avila Garcez et al. (2012)
1 Link prediction & symbolic vs. neural representations
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)

Tim Rocktäschel
End-to-End Differentiable Proving
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)
   - Fast but restricted (Demeester et al., 2016)

End-to-end differentiable proving (Rocktäschel and Riedel, 2017)
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)
   - Fast but restricted (Demeester et al., 2016)
   - Model-agnostic and fast (Minervini et al., 2017)
Outline

1 Link prediction & symbolic vs. neural representations
2 Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)
   - Fast but restricted (Demeester et al., 2016)
   - Model-agnostic and fast (Minervini et al., 2017)
3 End-to-end differentiable proving (Rocktäschel and Riedel, 2017)
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow \cite{rocktaschel2015}
   - Fast but restricted \cite{demeester2016}
   - Model-agnostic and fast \cite{minervini2017}
3. End-to-end differentiable proving \cite{rocktaschel2017}
   - Explicit multi-hop reasoning using neural networks
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)
   - Fast but restricted (Demeester et al., 2016)
   - Model-agnostic and fast (Minervini et al., 2017)
3. End-to-end differentiable proving (Rocktäschel and Riedel, 2017)
   - Explicit multi-hop reasoning using neural networks
   - Inducing rules using gradient descent
Outline

1. Link prediction & symbolic vs. neural representations
2. Regularize neural representations using logical rules
   - Model-agnostic but slow (Rocktäschel et al., 2015)
   - Fast but restricted (Demeester et al., 2016)
   - Model-agnostic and fast (Minervini et al., 2017)
3. End-to-end differentiable proving (Rocktäschel and Riedel, 2017)
   - Explicit multi-hop reasoning using neural networks
   - Inducing rules using gradient descent
4. Outlook & Summary
Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
Notation

- **Constant**: Homer, Bart, Lisa etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
  
  *Restricted to function-free terms in this talk*
Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
  - Restricted to function-free terms in this talk
- **Predicate**: fatherOf, parentOf etc.
  - function from terms to a Boolean
Notation

■ **Constant**: HOMER, BART, LISA etc. (lowercase)
■ **Variable**: X, Y etc. (uppercase, universally quantified)
■ **Term**: constant or variable
  Restricted to function-free terms in this talk
■ **Predicate**: fatherOf, parentOf etc.
  function from terms to a Boolean
■ **Atom**: predicate and terms, e.g., parentOf(X, BART)
Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
  
  *Restricted to function-free terms in this talk*

- **Predicate**: fatherOf, parentOf etc.
  
  function from terms to a Boolean

- **Atom**: predicate and terms, e.g., parentOf(X, BART)

- **Literal**: atom or negated or atom, e.g.,
  
  not parentOf(BART, LISA)
Notation

- **Constant**: HOMER, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
  
  *Restricted to function-free terms in this talk*
- **Predicate**: fatherOf, parentOf etc.
  
  function from terms to a Boolean
- **Atom**: predicate and terms, e.g., parentOf(X, BART)
- **Literal**: atom or negated or atom, e.g.,
  
  not parentOf(BART, LISA)
- **Rule**: head :- body.
  
  head: atom
  
  body: (possibly empty) list of literals representing conjunction
  
  *Restricted to Horn clauses in this talk*
Notation

- **Constant**: HOME, BART, LISA etc. (lowercase)
- **Variable**: X, Y etc. (uppercase, universally quantified)
- **Term**: constant or variable
  
  *Restricted to function-free terms in this talk*
- **Predicate**: fatherOf, parentOf etc.
  
  function from terms to a Boolean
- **Atom**: predicate and terms, e.g., parentOf(X, BART)
- **Literal**: atom or negated or atom, e.g.,
  
  not parentOf(BART, LISA)
- **Rule**: head :- body.
  
  head: atom
  
  body: (possibly empty) list of literals representing conjunction
  
  *Restricted to Horn clauses in this talk*
- **Fact**: ground rule (no free variables) with empty body, e.g.,
  
  parentOf(HOME, BART).
Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!
Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- \texttt{placeOfBirth} attribute is missing for 71\% of people!
Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- placeOfBirth attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- *placeOfBirth* attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts
Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- **placeOfBirth** attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts
Link Prediction

Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts

Das et al. (2017)
Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- **placeOfBirth** attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts

Das et al. (2017)
Real world knowledge bases (like Freebase, DBPedia, YAGO, etc.) are incomplete!

- `placeOfBirth` attribute is missing for 71% of people!
- Commonsense knowledge often not stated explicitly
- Weak logical relationships that can be used for inferring facts
Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  \[ \text{grandpaOf} \neq \text{grandfatherOf} \]

No notion of similarity:

\[ \text{apple} \sim \text{orange}, \text{professorAt} \sim \text{lecturerAt} \]

No generalization beyond what can be symbolically inferred:

\[ \text{isFruit}(\text{apple}), \text{apple} \sim \text{orange}, \text{isFruit}(\text{orange}) \]

Hard to work with language, vision and other modalities

"is a film based on the novel of the same name by" \( (X, Y) \)

But... leads to powerful inference mechanisms and proofs for predictions:

\[ \text{fatherOf}(\text{abe}, \text{homer}), \text{parentOf}(\text{homer}, \text{lisa}), \text{parentOf}(\text{homer}, \text{bart}) \]

\[ \text{grandfatherOf}(X,Y) :– \text{fatherOf}(X,Z), \text{parentOf}(Z,Y) \]

\[ \text{grandfatherOf}(\text{abe},Q) \] \[ \{ Q/\text{lisa}, Q/\text{bart} \} \]

Fairly easy to debug and trivial to incorporate domain knowledge:

Show to domain expert and let her change/add rules and facts
Symbolic Representations

- Symbols (constants and predicates) do not share any information: grandpaOf ≠ grandfather0f
- No notion of similarity: apple ∼ orange, professorAt ∼ lecturerAt

Hard to work with language, vision and other modalities

''is a film based on the novel of the same name by'' (X, Y)

But... leads to powerful inference mechanisms and proofs for predictions:

fatherOf(abe, homer), parentOf(homer, lisa), parentOf(homer, bart).

grandfatherOf(X, Y) :– fatherOf(X, Z), parentOf(Z, Y).

grandfatherOf(abe, Q) :– {Q/lisa}, {Q/bart}.

Fairly easy to debug and trivial to incorporate domain knowledge:

Show to domain expert and let her change/add rules and facts
Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  \[ \text{grandpaOf} \neq \text{grandfatherOf} \]

- No notion of similarity: \[ \text{APPLE} \sim \text{ORANGE}, \text{professorAt} \sim \text{lecturerAt} \]

- No generalization beyond what can be symbolically inferred:
  \[ \text{isFruit(A APPLE)}, \text{APPLE} \sim \text{ORGANGE}, \text{isFruit(ORANGE)}? \]
Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  \[\text{grandpaOf} \neq \text{grandfatherOf}\]
- No notion of similarity:
  \[\text{APPLE} \sim \text{ORANGE}, \text{professorAt} \sim \text{lecturerAt}\]
- No generalization beyond what can be symbolically inferred:
  \[\text{isFruit(APPLE), APPLE} \sim \text{ORGANGE, isFruit(ORANGE)}?\]
- Hard to work with language, vision and other modalities
  ‘‘is a film based on the novel of the same name by’’ \((X, Y)\)
Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  \[ \text{grandpaOf} \neq \text{grandfatherOf} \]

- No notion of similarity: \( \text{APPLE} \sim \text{ORANGE}, \text{professorAt} \sim \text{lecturerAt} \)

- No generalization beyond what can be symbolically inferred:
  \( \text{isFruit(APPLE), APPLE} \sim \text{ORGANAGE, isFruit(ORGANAGE)}? \)

- Hard to work with language, vision and other modalities
  ‘‘is a film based on the novel of the same name by’’(\(X, Y\))

- But... leads to powerful inference mechanisms and proofs for predictions:
  fatherOf(\(\text{ABE, HOMER}\)). parentOf(\(\text{HOMER, LISA}\)). parentOf(\(\text{HOMER, BART}\)).
  grandfatherOf(\(X, Y\)) :- fatherOf(\(X, Z\)), parentOf(\(Z, Y\)).
  grandfatherOf(\(\text{ABE, Q}\))? \{Q/LISA\}, \{Q/BART\}

Fairly easy to debug and trivial to incorporate domain knowledge:
Show to domain expert and let her change/add rules and facts
Symbolic Representations

- Symbols (constants and predicates) do not share any information:
  \[ \text{grandpaOf} \neq \text{grandfatherOf} \]

- No notion of similarity: \( \text{APPLE} \sim \text{ORANGE}, \text{professorAt} \sim \text{lecturerAt} \)

- No generalization beyond what can be symbolically inferred:
  \( \text{isFruit(APPLE), APPLE} \sim \text{ORGANGE, isFruit(ORANGE?)} \)

- Hard to work with language, vision and other modalities
  ‘‘is a film based on the novel of the same name by’’ \((X, Y)\)

- But... leads to powerful inference mechanisms and proofs for predictions:
  \( \text{fatherOf(ABE, HOMER). parentOf(HOMER, LISA). parentOf(HOMER, BART).} \)
  \( \text{grandfatherOf}(X, Y) :– \text{fatherOf}(X, Z), \text{parentOf}(Z, Y).} \)
  \( \text{grandfatherOf(ABE, Q)? \{Q/LISA\}, \{Q/BART\}} \)

- Fairly easy to debug and trivial to incorporate domain knowledge:
  Show to domain expert and let her change/add rules and facts
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  \[ \mathbf{v}_{\text{APPLE}}, \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{fatherOf}}, \ldots \in \mathbb{R}^k \]

Can capture similarity and even semantic hierarchy of symbols:
\[ \mathbf{v}_{\text{grandpaOf}} = \mathbf{v}_{\text{grandfatherOf}} , \quad \mathbf{v}_{\text{apple}} \sim \mathbf{v}_{\text{orange}} , \quad \mathbf{v}_{\text{apple}} < \mathbf{v}_{\text{fruit}} \]

Can be trained from raw task data (e.g. facts in a knowledge base)

Can be compositional
\[ \mathbf{v}_{\text{''is the father of''}} = \text{RNN}^\theta (\mathbf{v}_{\text{is}}, \mathbf{v}_{\text{the}}, \mathbf{v}_{\text{father}}, \mathbf{v}_{\text{of}}) \]

But... need large amount of training data

No direct way of incorporating prior knowledge

\[ \mathbf{v}_{\text{grandfatherOf}} (X, Y) :– \mathbf{v}_{\text{fatherOf}} (X, Z), \mathbf{v}_{\text{parentOf}} (Z, Y). \]
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  \[ \mathbf{v}_{\text{APPLE}}, \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{father0f}}, \ldots \in \mathbb{R}^k \]

- Can capture similarity and even semantic hierarchy of symbols:
  \[ \mathbf{v}_{\text{grandpa0f}} = \mathbf{v}_{\text{grandfather0f}}, \]
  \[ \mathbf{v}_{\text{APPLE}} \sim \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{APPLE}} < \mathbf{v}_{\text{FRUIT}} \]
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  \[ \mathbf{v}_{\text{APPLE}}, \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{fatherOf}}, \ldots \in \mathbb{R}^k \]

- Can capture similarity and even semantic hierarchy of symbols:
  \[ \mathbf{v}_{\text{grandpaOf}} = \mathbf{v}_{\text{grandfatherOf}}, \]
  \[ \mathbf{v}_{\text{APPLE}} \sim \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{APPLE}} < \mathbf{v}_{\text{FRUIT}} \]

- Can be trained from raw task data (e.g. facts in a knowledge base)

---

Tim Rocktäschel
End-to-End Differentiable Proving
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  \[ v_{\text{APPLE}}, v_{\text{ORANGE}}, v_{\text{father0f}}, \ldots \in \mathbb{R}^k \]

- Can capture similarity and even semantic hierarchy of symbols:
  \[ v_{\text{grandpa0f}} = v_{\text{grandfather0f}}, \]
  \[ v_{\text{APPLE}} \sim v_{\text{ORANGE}}, v_{\text{APPLE}} < v_{\text{FRUIT}} \]

- Can be trained from raw task data (e.g. facts in a knowledge base)

- Can be compositional
  \[ v^{\text{‘is the father of’}} = RNN_\theta(v_{\text{is}}, v_{\text{the}}, v_{\text{father}}, v_{\text{of}}) \]
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants): 
  \[ \mathbf{v}_{\text{apple}}, \mathbf{v}_{\text{orange}}, \mathbf{v}_{\text{fatherOf}}, \ldots \in \mathbb{R}^k \]

- Can capture similarity and even semantic hierarchy of symbols:
  \[ \mathbf{v}_{\text{grandpaOf}} = \mathbf{v}_{\text{grandfatherOf}}, \]
  \[ \mathbf{v}_{\text{apple}} \sim \mathbf{v}_{\text{orange}}, \mathbf{v}_{\text{apple}} < \mathbf{v}_{\text{fruit}} \]

- Can be trained from raw task data (e.g. facts in a knowledge base)

- Can be compositional
  \[ \mathbf{v}^{\text{‘is the father of’}} = \text{RNN}_\theta(\mathbf{v}_{\text{is}}, \mathbf{v}_{\text{the}}, \mathbf{v}_{\text{father}}, \mathbf{v}_{\text{of}}) \]

- But... need large amount of training data
Neural Representations

- Lower-dimensional fixed-length vector representations of symbols (predicates and constants):
  \[ \mathbf{v}_{\text{APPLE}}, \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{father0f}}, \ldots \in \mathbb{R}^k \]

- Can capture similarity and even semantic hierarchy of symbols:
  \[ \mathbf{v}_{\text{grandpa0f}} = \mathbf{v}_{\text{grandfather0f}}, \]
  \[ \mathbf{v}_{\text{APPLE}} \sim \mathbf{v}_{\text{ORANGE}}, \mathbf{v}_{\text{APPLE}} < \mathbf{v}_{\text{FRUIT}} \]

- Can be trained from raw task data (e.g. facts in a knowledge base)

- Can be compositional
  \[ \mathbf{v}^{\text{‘‘is the father of’’}} = \text{RNN}_\theta(\mathbf{v}_{\text{is}}, \mathbf{v}_{\text{the}}, \mathbf{v}_{\text{father}}, \mathbf{v}_{\text{of}}) \]

- But... need large amount of training data

- No direct way of incorporating prior knowledge
  \[ \mathbf{v}_{\text{grandfather0f}}(X, Y) :\leftarrow \mathbf{v}_{\text{father0f}}(X, Z), \mathbf{v}_{\text{parent0f}}(Z, Y). \]
State-of-the-art Neural Link Prediction

\[
livesIn(MELINDA, SEATTLE)? = f_\theta(v_{livesIn}, v_{MELINDA}, v_{SEATTLE})
\]
State-of-the-art Neural Link Prediction

\[
livesIn(\text{MELINDA, SEATTLE})? = f_{\theta}(v_{livesIn}, v_{\text{MELINDA}}, v_{\text{SEATTLE}})
\]

DistMult (Yang et al., 2015)

\[v_s, v_i, v_j \in \mathbb{R}^k\]
State-of-the-art Neural Link Prediction

\[ \text{livesIn(MELINDA, SEATTLE)?} = f_\theta(\text{livesIn}, \text{MELINDA}, \text{SEATTLE}) \]

**DistMult (Yang et al., 2015)**

\( \mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^k \)

\[
f_\theta(\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_s^\top (\mathbf{v}_i \odot \mathbf{v}_j)
= \sum_k \mathbf{v}_{sk} \mathbf{v}_{ik} \mathbf{v}_{jk}
\]
State-of-the-art Neural Link Prediction

\[ \text{livesIn}(\text{MELINDA, SEATTLE})? = f_\theta(\mathbf{v}_{\text{livesIn}}, \mathbf{v}_{\text{MELINDA}}, \mathbf{v}_{\text{SEATTLE}}) \]

**DistMult** *(Yang et al., 2015)*

\[ \mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^k \]

\[
\begin{align*}
    f_\theta(\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j) &= \mathbf{v}_s^T (\mathbf{v}_i \odot \mathbf{v}_j) \\
    &= \sum_k \mathbf{v}_{sk} \mathbf{v}_{ik} \mathbf{v}_{jk}
\end{align*}
\]

**ComplEx** *(Trouillon et al., 2016)*

\[ \mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j \in \mathbb{C}^k \]
State-of-the-art Neural Link Prediction

\[
\text{livesIn(MELINDA, SEATTLE)}? = f_\theta(\mathbf{v}_{\text{livesIn}}, \mathbf{v}_{\text{MELINDA}}, \mathbf{v}_{\text{SEATTLE}})
\]

**DistMult (Yang et al., 2015)**
\[
\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^k
\]
\[
f_\theta(\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_s^\top (\mathbf{v}_i \odot \mathbf{v}_j)
\]
\[
= \sum_k \mathbf{v}_{sk} \mathbf{v}_{ik} \mathbf{v}_{jk}
\]

**ComplEx (Trouillon et al., 2016)**
\[
\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j \in \mathbb{C}^k
\]
\[
f_\theta(\mathbf{v}_s, \mathbf{v}_i, \mathbf{v}_j) =
\]
\[
\begin{align*}
&\text{real}(\mathbf{v}_s)^\top \text{real}(\mathbf{v}_i) \odot \text{real}(\mathbf{v}_j) \\
&+ \text{real}(\mathbf{v}_s)^\top \text{imag}(\mathbf{v}_i) \odot \text{imag}(\mathbf{v}_j) \\
&+ \text{imag}(\mathbf{v}_s)^\top \text{real}(\mathbf{v}_i) \odot \text{imag}(\mathbf{v}_j) \\
&- \text{imag}(\mathbf{v}_s)^\top \text{imag}(\mathbf{v}_i) \odot \text{real}(\mathbf{v}_j)
\end{align*}
\]
State-of-the-art Neural Link Prediction

\[ \text{livesIn}(\text{MELINDA, SEATTLE})? = f_\theta(v_{\text{livesIn}}, v_{\text{MELINDA}}, v_{\text{SEATTLE}}) \]

**DistMult (Yang et al., 2015)**

\[ v_s, v_i, v_j \in \mathbb{R}^k \]

\[
f_\theta(v_s, v_i, v_j) = v_s^T (v_i \odot v_j)
= \sum_k v_{sk} v_{ik} v_{jk}
\]

**ComplEx (Trouillon et al., 2016)**

\[ v_s, v_i, v_j \in \mathbb{C}^k \]

\[
f_\theta(v_s, v_i, v_j) =
\begin{align*}
& \text{real}(v_s)^T (\text{real}(v_i) \odot \text{real}(v_j)) \\
& + \text{real}(v_s)^T (\text{imag}(v_i) \odot \text{imag}(v_j)) \\
& + \text{imag}(v_s)^T (\text{real}(v_i) \odot \text{imag}(v_j)) \\
& - \text{imag}(v_s)^T (\text{imag}(v_i) \odot \text{real}(v_j))
\end{align*}
\]

**Training Loss**

\[
\mathcal{L} = \sum_{r_s(e_i, e_j), y \in \mathcal{T}} -y \log (\sigma(f_\theta(v_s, v_i, v_j))) - (1 - y) \log (1 - \sigma(f_\theta(v_s, v_i, v_j)))
\]
State-of-the-art Neural Link Prediction

\[
livesIn(MELINDA, SEATTLE)? = f_\theta(v_{livesIn}, v_{MELINDA}, v_{SEATTLE})
\]

**DistMult** (Yang et al., 2015)

\[
v_s, v_i, v_j \in \mathbb{R}^k
\]

\[
f_\theta(v_s, v_i, v_j) = v_s^\top(v_i \circ v_j)
= \sum_k v_{sk} v_{ik} v_{jk}
\]

**ComplEx** (Trouillon et al., 2016)

\[
v_s, v_i, v_j \in \mathbb{C}^k
\]

\[
f_\theta(v_s, v_i, v_j) =
\begin{align*}
&\text{real}(v_s)^\top(\text{real}(v_i) \circ \text{real}(v_j)) + \text{real}(v_s)^\top(\text{imag}(v_i) \circ \text{imag}(v_j)) + \text{imag}(v_s)^\top(\text{real}(v_i) \circ \text{imag}(v_j)) - \text{imag}(v_s)^\top(\text{imag}(v_i) \circ \text{real}(v_j))
\end{align*}
\]

Training Loss

\[
\mathcal{L} = \sum_{r_s(e_i, e_j), y \in \mathcal{T}} -y \log(\sigma(f_\theta(v_s, v_i, v_j))) - (1 - y) \log(1 - \sigma(f_\theta(v_s, v_i, v_j)))
\]

- Learn \(v_s, v_i, v_j\) from data
State-of-the-art Neural Link Prediction

\[
livesIn(MELINDA, SEATTLE) = f_\theta(v_{livesIn}, v_{MELINDA}, v_{SEATTLE})
\]

**DistMult** *(Yang et al., 2015)*

\[
v_s, v_i, v_j \in \mathbb{R}^k
\]

\[
f_\theta(v_s, v_i, v_j) = v_s^T (v_i \odot v_j)
\]

\[
= \sum_k v_{sk} v_{ik} v_{jk}
\]

**ComplEx** *(Trouillon et al., 2016)*

\[
v_s, v_i, v_j \in \mathbb{C}^k
\]

\[
f_\theta(v_s, v_i, v_j) =
\]

\[
\text{real}(v_s)^T (\text{real}(v_i) \odot \text{real}(v_j))
\]

\[
+ \text{real}(v_s)^T (\text{imag}(v_i) \odot \text{imag}(v_j))
\]

\[
+ \text{imag}(v_s)^T (\text{real}(v_i) \odot \text{imag}(v_j))
\]

\[
- \text{imag}(v_s)^T (\text{imag}(v_i) \odot \text{real}(v_j))
\]

**Training Loss**

\[
\mathcal{L} = \sum_{r_s(e_i, e_j), y \in T} -y \log (\sigma(f_\theta(v_s, v_i, v_j))) - (1 - y) \log (1 - \sigma(f_\theta(v_s, v_i, v_j)))
\]

- Learn \(v_s, v_i, v_j\) from data
- Obtain gradients \(\nabla_{v_s} \mathcal{L}, \nabla_{v_i} \mathcal{L}, \nabla_{v_j} \mathcal{L}\) by backprop
Regularization by Propositional Logic

\[ \text{Link Predictor} \]

\[ u_1 \rightarrow \text{dot} \rightarrow \text{SIGM} \]
\[ u_2 \rightarrow \text{dot} \rightarrow \text{SIGM} \]
\[ u_3 \rightarrow \text{dot} \rightarrow \text{SIGM} \]

\[ p(F) = \begin{cases} f_{\theta}(s, i, j) & \text{if } F = s(i, j) \\ 1 - J & \text{if } F = \neg A \\ J & \text{if } F = A \land B \\ J & \text{if } F = A \lor B \\ (J - 1) + 1 & \text{if } F = A: \neg B \end{cases} \]

\[ \text{Loss} = -\log \left( \forall X, Y: f(X, Y) \right) = -\sum_{e_i, e_j \in C} \log f(e_i, e_j) \]

Rocktäschel et al. (2015), NAACL
Regularization by Propositional Logic

\[ \text{fatherOf}(X, Y) :\neg \text{parentOf}(X, Y), \neg \text{motherOf}(X, Y) \]
Regularization by Propositional Logic

\[
\text{fatherOf}(X, Y) :\neg \text{parentOf}(X, Y), \neg \text{motherOf}(X, Y)
\]

\[
p(F) = [F] = \begin{cases} 
  f_\theta(s, i, j) & \text{if } F = s(i, j) \\
  1 - [A] & \text{if } F = \neg A \\
  [A] [B] & \text{if } F = A \land B \\
  [A] + [B] - [A] [B] & \text{if } F = A \lor B \\
  [B] ([A] - 1) + 1 & \text{if } F = A :\neg B
\end{cases}
\]
Regularization by Propositional Logic

\[
\text{fatherOf}(X, Y) :\neg \text{parentOf}(X, Y), \neg \text{motherOf}(X, Y)
\]

\[
p(F) = [F] = \begin{cases} 
  f_\theta(s, i, j) & \text{if } F = s(i, j) \\
  1 - [A] & \text{if } F = \neg A \\
  [A] [B] & \text{if } F = A \land B \\
  [A] + [B] - [A] [B] & \text{if } F = A \lor B \\
  [B] ([A] - 1) + 1 & \text{if } F = A :\neg B 
\end{cases}
\]
Regularization by Propositional Logic

\[ p(F) = \begin{cases} 
  f_\theta(s, i, j) & \text{if } F = s(i, j) \\
  1 - [A] & \text{if } F = \neg A \\
  [A] [B] & \text{if } F = A \land B \\
  [A] + [B] - [A] [B] & \text{if } F = A \lor B \\
  [B] ([A] - 1) + 1 & \text{if } F = A \dashv B
\end{cases} \]

\[
\mathcal{L}(\{ \text{fatherOf}(\text{HOMER, BART}) \dashv \\
  \text{parentOf}(\text{HOMER, BART}) \land \\
  \neg \text{motherOf}(\text{HOMER, BART}) \})
\]

Rocktäschel et al. (2015), NAACL
Regularization by Propositional Logic

\[ \mathcal{L}(F) = - \log (\mathcal{L}(\forall X, Y : F(X, Y))) = - \sum_{(e_i, e_j) \in \mathcal{C}^2} \log \mathcal{L}[F(e_i, e_j)] \]

\[ p(F) = [F] = \begin{cases} 
  f_{\theta}(s, i, j) & \text{if } F = s(i, j) \\
  1 - [A] & \text{if } F = \neg A \\
  [A][B] & \text{if } F = A \land B \\
  [A] + [B] - [A][B] & \text{if } F = A \lor B \\
  [B]([A] - 1) + 1 & \text{if } F = A :\neg B 
\end{cases} \]

\[ \mathcal{L}(\forall \text{parentOf(HOMER, BART)} :\neg \text{motherOf(HOMER, BART)}]) \]

Rocktäschel et al. (2015), NAACL
Zero-shot Learning Results

Neural Link Prediction (LP)

Deduction after LP

Deduction before LP

Regularization

weighted Mean Average Precision

3
Zero-shot Learning Results

- Neural Link Prediction (LP)
- Deduction

Graph showing weighted Mean Average Precision with bars for Neural Link Prediction (LP) and Deduction.
Zero-shot Learning Results

- Neural Link Prediction (LP)
- Deduction
- Deduction after LP

![Bar Chart]

- Weighted Mean Average Precision

0 3 10 21

3 10 21
Zero-shot Learning Results

- Neural Link Prediction (LP)
- Deduction
- Deduction after LP
- Deduction before LP

Bar chart showing weighted Mean Average Precision with values:
- Neural Link Prediction (LP): 3
- Deduction: 10
- Deduction after LP: 21
- Deduction before LP: 33
Zero-shot Learning Results

- Neural Link Prediction (LP)
- Deduction
- Deduction after LP
- Deduction before LP
- Regularization

![Bar Chart](image)

- weighted Mean Average Precision

Values:
- Neural Link Prediction: 3
- Deduction: 10
- Deduction after LP: 21
- Deduction before LP: 33
- Regularization: 38
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent

\[ \forall X, Y: h(X, Y) :– b(X, Y) \]

\[ \forall (e_i, e_j) \in C_2: J h K \top J e_i, e_j K \geq J b K \top J e_i, e_j K \]

\[ \forall (e_i, e_j) \in C_2: J e_i, e_j K \in R_k^+ \]

Demeester et al. (2016), EMNLP
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent

∀ X, Y: h(X, Y) :– b(X, Y)
∀ (e_i, e_j) ∈ C_2: J_h K ⊤ J_e_i, e_j K ≥ J_b K ⊤ J_e_i, e_j K ≥ ∀ (e_i, e_j) ∈ C_2: J_e_i, e_j K ∈ R^{k+}
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent

Demeester et al. (2016), EMNLP
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent

Before

After
Lifted Regularization by Implications

Every father is a parent  Generalises to similar relations (e.g. dad)
Every mother is a parent Generalises to similar relations (e.g. mum)

Before

After

Demeester et al. (2016), EMNLP
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent
Every parent is a relative

Generalises to similar relations (e.g. dad)
Generalises to similar relations (e.g. mum)
No training facts needed!

Before

After

implied by father of

father of

mother of

parent of

relative of

father of

parent of

relative of

mother of

mum of

Demeester et al. (2016), EMNLP
Lifted Regularization by Implications

Every father is a parent
Every mother is a parent
Every parent is a relative

∀X, Y : h(X, Y) := b(X, Y)

∀(e_i, e_j) ∈ C^2 : [h]_i^T [e_i, e_j] ≥ [b]_i^T [e_i, e_j]

[h] ≥ [b], ∀(e_i, e_j) ∈ C^2 : [e_i, e_j] ∈ ℜ^k_+

Demeester et al. (2016), EMNLP
Adversarial Regularization

Clause $A$: $h(X, Y) : \neg b_1(X, Z) \land b_2(Z, Y)$

- Regularization by propositional rules needs grounding – does not scale to large domains!
Adversarial Regularization

Clause $\mathcal{A}$: $h(X, Y) := b_1(X, Z) \land b_2(Z, Y)$

- Regularization by propositional rules needs grounding – does not scale to large domains!
- Lifted regularization only supports direct implications

Minervini et al. (2017), UAI
Adversarial Regularization

Clause \( A: \quad h(X, Y) :\neg b_1(X, Z) \land b_2(Z, Y) \)

- Regularization by propositional rules needs grounding – does not scale to large domains!
- Lifted regularization only supports direct implications
- Idea: let grounding be generated by an adversary and optimize minimax game...

Minervini et al. (2017), UAI
Adversarial Regularization

Clause $A$: $h(X, Y) : - b_1(X, Z) \land b_2(Z, Y)$

- Regularization by propositional rules needs grounding – does not scale to large domains!
- Lifted regularization only supports direct implications
- Idea: let grounding be generated by an adversary and optimize minimax game...
- Adversary finds maximally violating grounding for a given rule

Inconsistency Loss

Minervini et al. (2017), UAI
Adversarial Regularization

Clause $A$: \[ h(X, Y) :\leftarrow b_1(X, Z) \land b_2(Z, Y) \]

- Regularization by propositional rules needs grounding – does not scale to large domains!
- Lifted regularization only supports direct implications
- Idea: let grounding be generated by an adversary and optimize minimax game...
- Adversary finds maximally violating grounding for a given rule
- Neural link predictor attempts to minimize rule violation for given generated groundings

Minervini et al. (2017), UAI
End-to-End Differentiable Prover

- Neural network for proving queries to a knowledge base
End-to-End Differentiable Prover

- Neural network for proving queries to a knowledge base
- Proof success differentiable w.r.t. vector representations of symbols
End-to-End Differentiable Prover

- Neural network for proving queries to a knowledge base
- Proof success differentiable w.r.t. vector representations of symbols
- **Learn vector representations of symbols** end-to-end from proof success
End-to-End Differentiable Prover

- Neural network for proving queries to a knowledge base
- Proof success differentiable w.r.t. vector representations of symbols
- **Learn vector representations of symbols** end-to-end from proof success
- **Make use of provided rules** in soft proofs
End-to-End Differentiable Prover

- Neural network for proving queries to a knowledge base
- Proof success differentiable w.r.t. vector representations of symbols
- **Learn vector representations of symbols** end-to-end from proof success
- **Make use of provided rules** in soft proofs
- **Induce interpretable rules** end-to-end from proof success
Approach

Let's neuralize Prolog's Backward Chaining using a Radial Basis Function kernel for unifying vector representations of symbols!

Nando de Freitas @NandoDF · 5 Aug 2016
Neuralise (verb, #neuralize): to implement a known thing with deep nets. Usage: Let's neuralize warping, neuralize this! And train it!
Approach

Nando de Freitas @NandoDF · 5 Aug 2016
Neuralise (verb, #neuralize): to implement a known thing with deep nets. Usage:
Let's neuralize warping, neuralize this! And train it!

Yann LeCun @ylecun
Replying to @NandoDF

sort of like "kernelize" used to be.

10:11 AM · 5 Aug 2016
Approach

Let's neuralize Prolog's Backward Chaining using a Radial Basis Function kernel for unifying vector representations of symbols!

Let’s **neuralize** Prolog’s Backward Chaining using a Radial Basis Function **kernel** for unifying vector representations of symbols!
Prolog’s Backward Chaining

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).
Prolog’s Backward Chaining

**Example Knowledge Base:**

1. `fatherOf(ABE, HOMER).`
2. `parentOf(HOMER, BART).`
3. `grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).`

**Intuition:**

- Backward chaining translates a query into subqueries via rules, e.g.,
  
  `grandfatherOf(ABE, BART) \Rightarrow 3. fatherOf(ABE, Z), parentOf(Z, BART)`
Prolog’s Backward Chaining

Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Intuition:

- Backward chaining translates a query into subqueries via rules, e.g.,
  grandfatherOf(ABE, BART) \[\Rightarrow\] fatherOf(ABE, Z), parentOf(Z, BART)
- It attempts this for all rules in the knowledge base and thus specifies a depth-first search
Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Query

grandfatherOf ABE BART
Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Query

grandfatherOf ABE BART

1. fatherOf ABE HOMER
Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Query

grandfatherOf ABE BART

fatherOf ABE HOMER
Example Knowledge Base:

1. \texttt{fatherOf(ABE, HOMER)}.
2. \texttt{parentOf(HOMER, BART)}.
3. \texttt{grandfatherOf(X, Y) :- fatherOf(X, Z), parentOf(Z, Y)}.

Query

\texttt{grandfatherOf(ABE, BART)}

\[ ? = ? = ? \]

1. \texttt{fatherOf(ABE, HOMER)}

\[ \text{FAIL} \quad \text{SUCCESS} \quad \text{FAIL} \]
Example Knowledge Base:
1. \texttt{fatherOf(ABE, HOMER)}.
2. \texttt{parentOf(HOMER, BART)}.
3. \texttt{grandfatherOf(X, Y) :- fatherOf(X, Z),
parentOf(Z, Y)}.

Query
\texttt{grandfatherOf} ABE BART

State \( t \)
\[
\text{\( \emptyset \)} \quad \text{SUCCESS}
\]

1. \texttt{fatherOf} ABE HOMER

\[
? \quad ? \quad ?
\]

\[
\text{FAIL} \quad \text{SUCCESS} \quad \text{FAIL}
\]
Unification

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Query

grandfatherOf(ABE, BART)

State $t$

$\emptyset$ SUCCESS

1. fatherOf(ABE, HOMER) FAIL $\emptyset$ SUCCESS $\emptyset$ FAIL

FAIL SUCCESS FAIL
Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Query
grandfatherOf(ABE, BART)

State $t$

\[ \emptyset \rightarrow \text{SUCCESS} \]

\[ 2. \text{parentOf} \rightarrow \text{FAIL} \]

\[ ? = ? = ? \]

State $t + 1$

\[ \emptyset \rightarrow \text{FAIL} \]

FAIL \hspace{1cm} FAIL \hspace{1cm} SUCCESS
Example Knowledge Base:
1. \texttt{fatherOf}(\texttt{ABE, HOMER}).
2. \texttt{parentOf}(\texttt{HOMER, BART}).
3. \texttt{grandfatherOf}(\texttt{X, Y}) :-
   \texttt{fatherOf}(\texttt{X, Z}),
   \texttt{parentOf}(\texttt{Z, Y}).

Query
\texttt{grandfatherOf}(\texttt{ABE, BART})

State $t$
\[ \emptyset \]
\texttt{SUCCESS}

State $t + 1$
\[ \texttt{X/ABE} \]
\[ \texttt{Y/BART} \]
\texttt{SUCCESS}
Unification Failure

Example Knowledge Base:
1. \( \text{fatherOf}(\text{ABE}, \text{HOMER}). \)
2. \( \text{parentOf}(\text{HOMER}, \text{BART}). \)
3. \( \text{grandfatherOf}(X, Y) :- \text{fatherOf}(X, Z), \text{parentOf}(Z, Y). \)

Query

\( \text{grandpaOf}(\text{ABE}, \text{BART}). \)

State \( t \)

\( \emptyset \) SUCCESS

\( \emptyset \) SUCCESS

3. \( \text{grandfatherOf}(X, Y) \) FAIL

X/ABE Y/BART FAIL

State \( t + 1 \)
Neural Unification

Example Knowledge Base:

1. \text{fatherOf}(\text{ABE, HOMER}).
2. \text{parentOf}(\text{HOMER, BART}).
3. \text{grandfatherOf}(X, Y) :- \\
   \text{fatherOf}(X, Z), \\
   \text{parentOf}(Z, Y).

Query

\text{grandfatherOf}(\text{ABE, BART}).

State $t$

\[ \emptyset \text{ 1.0} \]

State $t+1$

\[ \text{X/ABE, Y/BART} \]
Neural Unification

Example Knowledge Base:
1. \text{fatherOf}(\text{ABE}, \text{HOMER}).
2. \text{parentOf}(\text{HOMER}, \text{BART}).
3. \text{grandfatherOf}(X, Y) :-
   \text{fatherOf}(X, Z),
   \text{parentOf}(Z, Y).

Query

\text{grandfatherOf}(\text{ABE}, \text{BART}).

\text{grandpaOf}(\text{ABE}, \text{BART}).

\begin{align*}
\min \left(1.0, \exp \left(-\frac{\|v_{\text{grandpaOf}} - v_{\text{grandfatherOf}}\|_2}{2\mu^2}\right)\right)
\end{align*}
Example Knowledge Base:

1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :- fatherOf(X, Z), parentOf(Z, Y).

Differentiable Prover

∅ ; 1.0

grandpaOf ABE BART
Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅; 1.0

∅; 1.0

grandpaOf

ABE

BART

∅; 1.0
Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

Differentiable Prover
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅ ; 1.0

grandpaOf
ABE

grandfatherOf(X, Y) :-
fatherOf(X, Z),
parentOf(Z, Y).

X/ABE

Y/BART

3.1 fatherOf(X, Z)
3.2 parentOf(Z, Y)
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅ : 1.0

grandpaOf ABE BART

∅ : 1.0

X/ABE Y/BART

fatherOf ABE Z

3.1 fatherOf(X, Z)
3.2 parentOf(Z, Y)

FAIL

FAIL

FAIL

FAIL
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅ : 1.0

FAIL
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :- fatherOf(X, Z), parentOf(Z, Y).

∅ 1.0

∅ : 1.0

grandpaOf ABE BART

3.1 fatherOf(X, Z)
3.2 parentOf(Z, Y)

fatherOf ABE

3.2 parentOf(Z, Y)

X/ABE Y/BART
Z/HOMER

FAIL

FAIL
Example Knowledge Base:

1. \text{fatherOf}(\text{ABE}, \text{HOMER}).
2. \text{parentOf}(\text{HOMER}, \text{BART}).
3. \text{grandfatherOf}(X, Y) :– \text{fatherOf}(X, Z), \text{parentOf}(Z, Y).

\begin{align*}
\emptyset & : 1.0 \\
\text{grandpaOf} & : \text{ABE, BART} \\
\text{fatherOf} & : \text{ABE} \\
\text{parentOf} & : \text{HOMER, BART}
\end{align*}
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :- fatherOf(X, Z), parentOf(Z, Y).

∅; 1.0

grandpaOf

fatherOf

parentOf

FAIL

FAIL

FAIL
Differentiable Prover

Example Knowledge Base:

1. fatherOf(abe, homer).
2. parentOf(homer, bart).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅ 1.0

∅

∅

∅

∅ 1.0

grandpaOf ABE BART

fatherOf ABE Z

parentOf Z Y

FAIL

FAIL

FAIL

FAIL
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

\[
\emptyset \\
1.0
\]

Tim Rocktäschel
End-to-End Differentiable Proving
Differentiable Prover

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅

1.0
Neural Program Induction

Example Knowledge Base:
1. fatherOf(ABE, HOMER).
2. parentOf(HOMER, BART).
3. grandfatherOf(X, Y) :-
   fatherOf(X, Z),
   parentOf(Z, Y).

∅ :– 1.0

∅

fatherOf  ABE  BART

∅

grandpaOf

X/ABE  Y/BART

X/ABE

Y/BART

Z/HOMER

FAIL

FAIL

FAIL

FAIL

FAIL

FAIL
Example Knowledge Base:
1. fatherOf(abe, homer).
2. parentOf(homer, bart).
3. \( \theta_1(X, Y) := \theta_2(X, Z), \theta_3(Z, Y) \).
Training Objective

\[
\theta \left( \text{grandpaOf}(\text{abe}, \text{bart}) \right)
\]

Max pooling

Loss: negative log-likelihood w.r.t. target proof success

Trained end-to-end using stochastic gradient descent

Vectors are learned such that proof success is high for known facts and low for sampled negative facts

Tim Rocktäschel
End-to-End Differentiable Proving
Training Objective

\[ f_\theta(\text{grandpaOf}(\text{ABE}, \text{BART})) \]

Max pooling

\[
\text{Loss: negative log-likelihood w.r.t. target proof success}
\]

Trained end-to-end using stochastic gradient descent

Vectors are learned such that proof success is high for known facts and low for sampled negative facts.
Training Objective

\[ f_\theta(\text{grandpaOf}(\text{ABE}, \text{BART})) \]

- Loss: negative log-likelihood w.r.t. target proof success
Training Objective

- Loss: negative log-likelihood w.r.t. target proof success
- Trained end-to-end using stochastic gradient descent
Training Objective

- Loss: negative log-likelihood w.r.t. target proof success
- Trained end-to-end using stochastic gradient descent
- Vectors are learned such that proof success is high for known facts and low for sampled negative facts
Calculation on GPU
Calculation on GPU
Calculation on GPU

Diagram showing the calculation process with nodes labeled as fatherOf, parentOf, grandchildOf, idiotOf, and others. The diagram illustrates the process of unifying symbols with a question mark at the end.
Calculation on GPU

Q

fatherOf

parentOf

grandmaOf

fatherOf

parentOf

grandmaOf

unify

unify (symbolic)

HOMER

BART

LISA

HOMER

BART

LISA

HOMER

BART

LISA

HOMER

BART

LISA

HOMER

BART

LISA
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)
Experiments

Benchmark Knowledge Bases: **Kinship**, **Nations**, **UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)
Experiments

Benchmark Knowledge Bases: **Kinship, Nations, UMLS** (Kok and Domingos, 2007), and **Countries** (Bouchard et al., 2015)
Details

- Models implemented in TensorFlow
Details

- Models implemented in TensorFlow
  - **ComplEx** Neural link prediction model by Trouillon et al. (2016)
Details

- Models implemented in TensorFlow
  - ComplEx  Neural link prediction model by Trouillon et al. (2016)
  - Prover  End-to-end differentiable prover
Models implemented in TensorFlow

- **ComplEx** Neural link prediction model by Trouillon et al. (2016)
- **Prover** End-to-end differentiable prover
- **Proverλ** Same, but representations trained with ComplEx as auxiliary task
Details

- Models implemented in TensorFlow
  - ComplEx  Neural link prediction model by Trouillon et al. (2016)
  - Prover  End-to-end differentiable prover
  - Proverλ Same, but representations trained with ComplEx as auxiliary task

- Rule Templates:

  **Kinship, Nations & UMLS**
  20 #1(X, Y) :- #2(X, Y).
  20 #1(X, Y) :- #2(Y, X).
  20 #1(X, Y) :- #2(X, Z), #3(Z, Y).

  **Countries S1**
  3 #1(X, Y) :- #1(Y, X).
  3 #1(X, Y) :- #2(X, Z), #2(Z, Y).

  **Countries S2**
  3 #1(X, Y) :- #2(X, Z), #3(Z, Y).

  **Countries S3**
  3 #1(X, Y) :- #2(X, Z), #3(Z, W), #4(W, Y).
Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ComplEx</th>
<th>Prover</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries S3</td>
<td>48</td>
<td>57</td>
</tr>
<tr>
<td>Kinship</td>
<td>48</td>
<td>70</td>
</tr>
<tr>
<td>Nations</td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>UML</td>
<td>82</td>
<td>82</td>
</tr>
</tbody>
</table>

The chart shows the accuracy (HITS@1) for different datasets using the ComplEx and Prover models.
Results

<table>
<thead>
<tr>
<th></th>
<th>ComplEx</th>
<th>Prover</th>
<th>Proverλ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries</td>
<td>48</td>
<td>57</td>
<td>77</td>
</tr>
<tr>
<td>S3</td>
<td>70</td>
<td>76</td>
<td>82</td>
</tr>
<tr>
<td>Kinship</td>
<td>48</td>
<td>59</td>
<td>82</td>
</tr>
<tr>
<td>Nations</td>
<td>62</td>
<td>62</td>
<td>87</td>
</tr>
<tr>
<td>UMLS</td>
<td>87</td>
<td>82</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy / HITS@1
## Examples of Induced Rules

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Induced rules and their confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>$\text{locatedIn}(X, Y) \leftarrow \text{locatedIn}(X, Z), \text{locatedIn}(Z, Y)$.</td>
</tr>
<tr>
<td>S2</td>
<td>$\text{locatedIn}(X, Y) \leftarrow \text{neighborOf}(X, Z), \text{locatedIn}(Z, Y)$.</td>
</tr>
<tr>
<td>S3</td>
<td>$\text{locatedIn}(X, Y) \leftarrow \text{neighborOf}(X, Z), \text{neighborOf}(Z, W), \text{locatedIn}(W, Y)$.</td>
</tr>
<tr>
<td>Nations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{blockpositionindex}(X, Y) \leftarrow \text{blockpositionindex}(Y, X)$.</td>
</tr>
<tr>
<td></td>
<td>$\text{expeldiplomats}(X, Y) \leftarrow \text{negativebehavior}(X, Y)$.</td>
</tr>
<tr>
<td></td>
<td>$\text{negativecomm}(X, Y) \leftarrow \text{commonbloc0}(X, Y)$.</td>
</tr>
<tr>
<td></td>
<td>$\text{intergovorgs3}(X, Y) \leftarrow \text{intergovorgs}(Y, X)$.</td>
</tr>
<tr>
<td>UMLS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{interacts}<em>{\text{with}}(X, Y) \leftarrow \text{interacts}</em>{\text{with}}(X, Z), \text{interacts}_{\text{with}}(Z, Y)$.</td>
</tr>
<tr>
<td></td>
<td>$\text{isa}(X, Y) \leftarrow \text{isa}(X, Z), \text{isa}(Z, Y)$.</td>
</tr>
<tr>
<td></td>
<td>$\text{derivative}<em>{\text{of}}(X, Y) \leftarrow \text{derivative}</em>{\text{of}}(X, Z), \text{derivative}_{\text{of}}(Z, Y)$.</td>
</tr>
</tbody>
</table>
My patient is not responding after three days of codeine treatment. What could have happened?

Answer: Morphine intoxication

Proof:
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
Question

My patient is not responding after three days of codeine treatment. What could have happened?

Answer

- Morphine intoxication
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
My patient is not responding after three days of codeine treatment. What could have happened?

**Answer**
Morphine intoxication

- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
Question
My patient is not responding after three days of codeine treatment. What could have happened?

Morphine intoxication
Proof
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
Question
My patient is not responding after three days of codeine treatment. What could have happened?

Answer
Morphine intoxication
Proof
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
My patient is not responding after three days of codeine treatment. What could have happened?

Answer
- Morphine intoxication
  - Codeine is metabolized to morphine
  - Mutation in CYP2D6 can cause ultrarapid metabolization
  - Ultrarapid metabolization can lead to morphine overdose
  - Morphine overdose is an intoxication
My patient is not responding after three days of codeine treatment. What could have happened?

Answer
Morphine intoxication

Proof
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
My patient is not responding after three days of codeine treatment. What could have happened?

Answer
Morphine intoxication

Proof
- Codeine is metabolized to morphine
- Mutation in CYP2D6 can cause ultrarapid metabolization
- Ultrarapid metabolization can lead to morphine overdose
- Morphine overdose is an intoxication
Summary

- We proposed various ways of \textit{regularizing vector representations of symbols using rules}. 

Future research:
- Scaling up to larger knowledge bases
- Connecting to RNNs for proving with natural language statements
Summary

- We proposed various ways of regularizing vector representations of symbols using rules.
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.
Summary

- We proposed various ways of regularizing vector representations of symbols using rules
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base
- Proof success differentiable w.r.t. vector representations of symbols

Future research:
- Scaling up to larger knowledge bases
- Connecting to RNNs for proving with natural language statements
We proposed various ways of regularizing vector representations of symbols using rules.

We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.

Proof success differentiable w.r.t. vector representations of symbols.

Symbolic rule application but neural unification.

Future research:
- Scaling up to larger knowledge bases.
- Connecting to RNNs for proving with natural language statements.
We proposed various ways of regularizing vector representations of symbols using rules.

We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.

Proof success differentiable w.r.t. vector representations of symbols.

Symbolic rule application but neural unification.

Learns vector representations of symbols from data via gradient descent.
Summary

- We proposed various ways of regularizing vector representations of symbols using rules.
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.
- Proof success differentiable w.r.t. vector representations of symbols.
- Symbolic rule application but neural unification.
- Learns vector representations of symbols from data via gradient descent.
- Induces interpretable rules from data via gradient descent.
Summary

- We proposed various ways of regularizing vector representations of symbols using rules.
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.
- Proof success differentiable w.r.t. vector representations of symbols.
- Symbolic rule application but neural unification.
- Learns vector representations of symbols from data via gradient descent.
- Induces interpretable rules from data via gradient descent.
- Various computational optimizations: batch proving, tree pruning etc.

Future research:
- Scaling up to larger knowledge bases.
- Connecting to RNNs for proving with natural language statements.
Summary

- We proposed various ways of regularizing vector representations of symbols using rules.
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.
- Proof success differentiable w.r.t. vector representations of symbols.
- **Symbolic rule application but neural unification.**
- **Learns vector representations of symbols** from data via gradient descent.
- **Induces interpretable rules** from data via gradient descent.
- Various computational optimizations: batch proving, tree pruning etc.
- Future research:
We proposed various ways of regularizing vector representations of symbols using rules.

We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.

Proof success differentiable w.r.t. vector representations of symbols.

**Symbolic rule application but neural unification**

Learns vector representations of symbols from data via gradient descent.

Induces interpretable rules from data via gradient descent.

Various computational optimizations: batch proving, tree pruning etc.

Future research:

- Scaling up to larger knowledge bases.
Summary

- We proposed various ways of regularizing vector representations of symbols using rules.
- We used Prolog’s backward chaining as recipe for recursively constructing a neural network to prove queries to a knowledge base.
- Proof success differentiable w.r.t. vector representations of symbols.
- **Symbolic rule application but neural unification**
- **Learns vector representations of symbols** from data via gradient descent.
- **Induces interpretable rules** from data via gradient descent.
- Various computational optimizations: batch proving, tree pruning etc.
- Future research:
  - **Scaling up** to larger knowledge bases.
  - **Connecting to RNNs** for proving with natural language statements.
Thank you!

http://rockt.github.com
tim.rocktaschel@cs.ox.ac.uk
Twitter: @_rockt
References I


