Inductive Logic Programming
Lecture 2.1
Meta-Interpretive Learning and Bias Reformulation

Stephen Muggleton
Department of Computing
Imperial College, London

5th July, 2018

Paper for this lecture

Paper05: D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, and S.H. Muggleton. Bias reformulation for one-shot function induction. In Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014), pages 525-530, Amsterdam, 2014. IOS Press.

Textual analogy problem

bob	BOB
alice	?

Alternative 1

bob	BOB
alice	BOICE

Harder tasks

Task1	miKe dwIGHT	Mike Dwight
Task2	European Conference on Artificial Intelligence	ECAI
Task3	My name is John.	John

Question

How can this human text transformation bias be learned by a computer?

Option1: Hardcode bias as DSL [Gulwani 2011,2012]
FlashFill, Excel 2013

brent.harold@hotmail.com matthew.rosman@yahoo.com jim.james@fas.harvard.edu ruby.clinton@mit.edu josh.smith@gmail.com	Brent Harold

Option1: Hardcode the bias [Gulwani 2011,2012]
FlashFill, Excel 2013

brent.harold@hotmail.com	Brent Harold
matthew.rosman@yahoo.com	Matthew Rosman
jim.james@fas.harvard.edu	Jim James
ruby.clinton@mit.edu	Ruby Clinton
josh.smith@gmail.com	Josh Smith

Option1: Non-intuitive FlashFill error

IaN RoDny	Ian Rodny
StaNleY TRAVis	Itanley Rley travis

Option2: Learn bias using variant of Meta-Interpretive Learning (IJCAI 2013)

brent.harold@hotmail.com	Brent Harold

ep04(A,B) ← ep04_1(A,C), ep04_2(C,B).

ep04_1(A,B) ← ep04_3(A,C), ep04_4(C,B).

ep04_2(A,B) ← ep04_3(A,C), skiprest(C,B).

ep04_3(A,B) ← make_uppercase(A,C), copyword(C,B).

ep04_4(A,B) ← skip1(A,C), write_space(C,B).

Learning time = **9.3 seconds**

Sequential episodes

ep02	james	James.

ep04	brent.harold@hotmail.com	Brent Harold

ep02(A,B) ← ep02_1(A,C), write_dot(C,B).

ep02_1(A,B) ← make_upper(A,C), copyword(C,B).

ep04(A,B) :- ep04_2(A,C), ep04_3(C,B).

ep04_2(A,B) :- ep04_4(A,C), skip1(C,B).

ep04_3(A,B) :- ep02_1(A,C), skiprest(C,B).

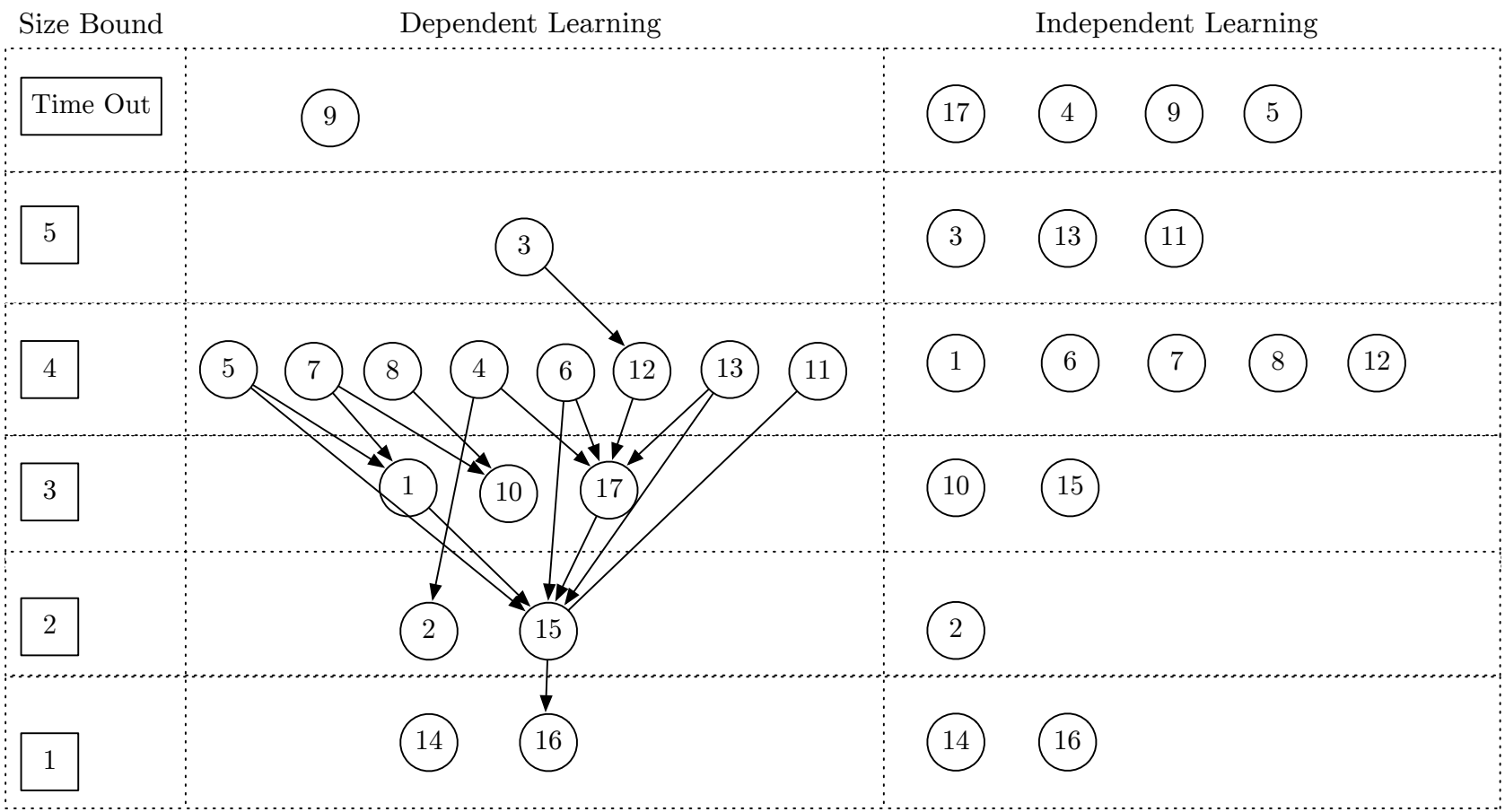
ep04_4(A,B) :- ep02_1(A,C), write_space(C,B).

Learning time = **3.1 seconds**

Language class and complexity

- Datalog hypothesis class H^2_2 maximum arity two and two atoms in body.
- Cardinality of hypothesis space $O(m^n p^{3n})$ given m metarules, n clauses and p predicate symbols.

Dependent vs Independent Learning



Chain of programs from dependent learning

f_{03}(A,B) :- f_{12_1}(A,C), f_{12}(C,B).

f_{12}(A,B) :- f_{12_1}(A,C), f_{12_2}(C,B).

f_{12_1}(A,B) :- f_{12_2}(A,C), $skip1$(C,B).

f_{12_2}(A,B) :- f_{12_3}(A,C), $write1$(C,B,'.').

f_{12_3}(A,B) :- $copy1$(A,C), f_{17_1}(C,B).

f_{17}(A,B) :- f_{17_1}(A,C), f_{15}(C,B).

f_{17_1}(A,B) :- f_{15_1}(A,C), f_{17_1}(C,B).

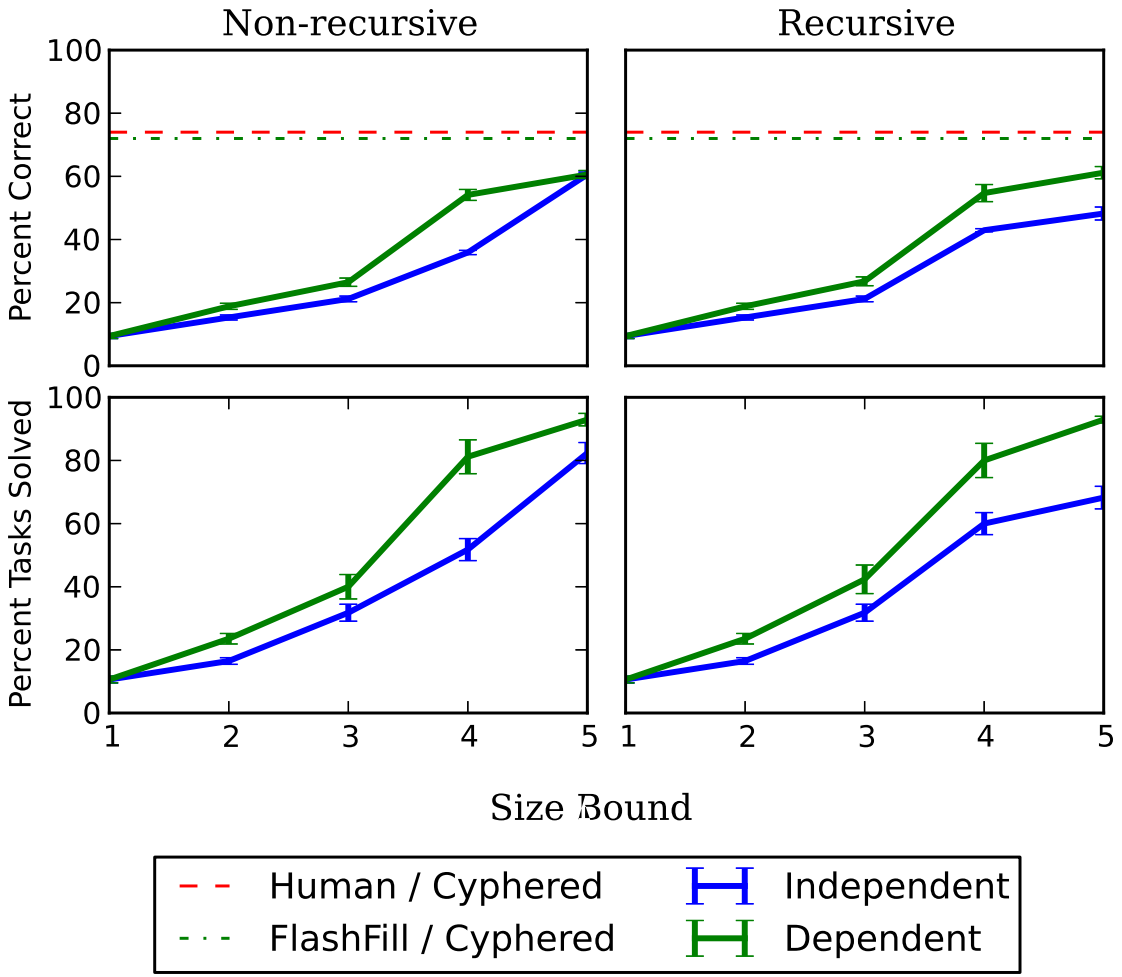
f_{17_1}(A,B) :- $skipalphanum$(A,B).

f_{15}(A,B) :- f_{15_1}(A,C), f_{16}(C,B).

f_{15_1}(A,B) :- $skipalphanum$(A,C), $skip1$(C,B).

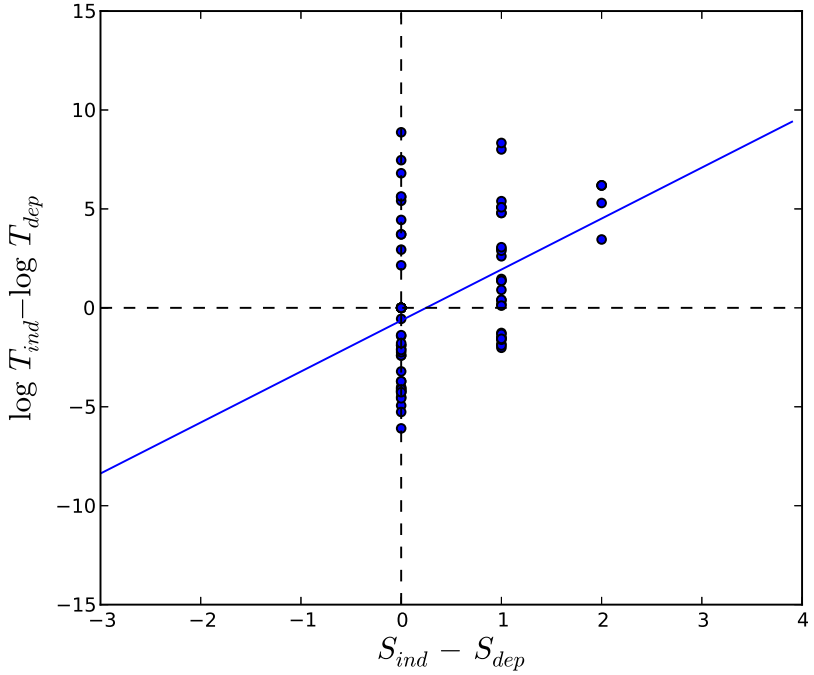
f_{16}(A,B) :- $copyalphanum$(A,C), $skiprest$(C,B).

Performance graphs

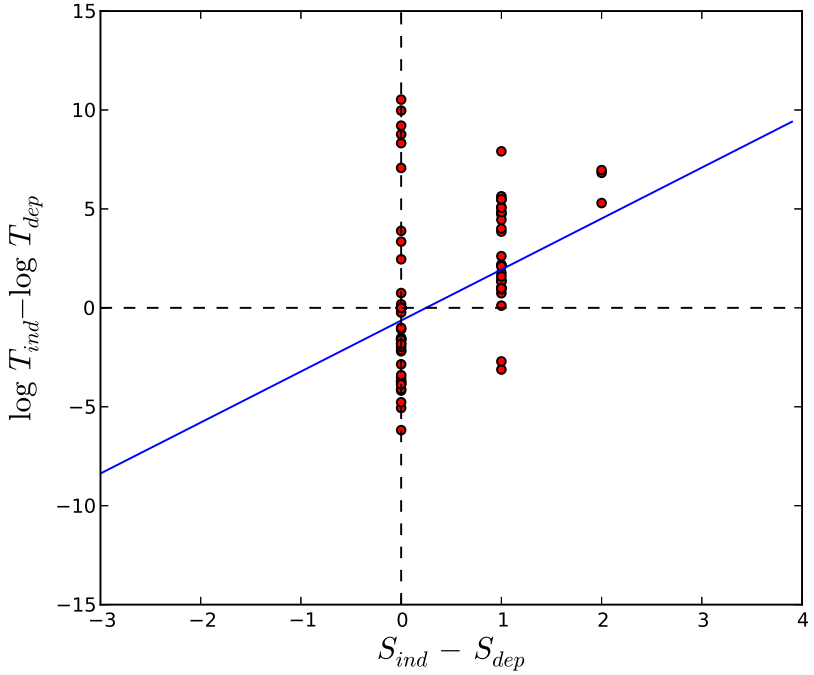


Running times Dependent vs Independent

No recursion



Recursion



Summary and limitations

- Multi-task learning from one example
- Meta-interpretive learning - efficient predicate invention and learning recursion
- Simple, compact and general approach to domain specific inductive programming
- Design of better intelligent interfaces
- Need to look at larger sets of transformation tasks
- Use of push and pop operations
- Need ways of handling noisy examples