Inductive Programming
Lecture 1
End-User Programming by Induction

Stephen Muggleton
University of Nanjing and
Emeritus Professor Imperial College London

9th October, 2025

Overview of Inductive Programming

Each **Lecture** immediately followed by **Tutorial**.

Lecture 1	09-10-25	End-user Programming by Induction
Lecture 2	14-10-25	Domain-specific languages and Background Knowledge
Lecture 3	16-10-25	One-shot induction and Bias reformulation
Lecture 4	21-10-25	Inducing an Algorithm from One Example
Lecture 5	23-10-25	Induction of Efficient Programs
Lecture 6	27-10-25	Comprehensibility
Lecture 7	28-10-25	Data wrangling
Lecture 8	30-10-25	Game Strategy Induction

Lecture material

Lecture material:

```
http://www.doc.ic.ac.uk/~shm/IP/Lecture1.pdf
http://www.doc.ic.ac.uk/~shm/IP/Lecture2.pdf
...
```

Presentation of IP course

- Research papers provided for each lecture in place of lecture notes
- Tutorial sheets provided with model answers

Paper for this lecture

Paper1.1: S. Gulwani, J. Hernandez-Orallo, E. Kitzelmann, S.H. Muggleton, U. Schmid, and B. Zorn. Inductive programming meets the real world. Communications of the ACM, 58(11):90-99, 2015.

Motivation - End-User Programming

- Much of world population use computers for everyday tasks
- Most end-users cannot program
- Often perform repetitive tasks manually
- Programming by example - Inductive Programming - Mass Market? - Microsoft Excel 2013- release of FlashFill
- Small but complex programs induced from few examples

FlashFill (Excel 2013, Gulwani, ACM Milner award 2014)

	A	B
1	Email	Column 2
2	Nancy.FreeHafer@fourthcoffee.com	nancy freehafer
3	Andrew.Cencici@northwindtraders.com	andrew cencici
4	Jan.Kotas@litwareinc.com	jan kotas
5	Mariya.Sergienko@gradicdesigninstitute.com	mariya sergienko
6	Steven.Thorpe@northwindtraders.com	steven thorpe
7	Michael.Neipper@northwindtraders.com	michael neipper
8	Robert.Zare@northwindtraders.com	robert zare
9	Laura.Giussani@adventure-works.com	laura giussani
10	Anne.HL@northwindtraders.com	anne hl
11	Alexander.David@contoso.com	alexander david
12	Kim.Shane@northwindtraders.com	kim shane
13	Manish.Chopra@northwindtraders.com	manish chopra
14	Gerwald.Oberleitner@northwindtraders.com	gerwald oberleitner
15	Amr.Zaki@northwindtraders.com	amr zaki
16	Yvonne.McKay@northwindtraders.com	yvonne mckay
17	Amanda.Pinto@northwindtraders.com	amanda pinto

Induced string transformation program

$$\mathit{Concatenate}(\mathit{ToLower}(\mathit{Substring}(v, \mathit{WordToken}, 1)),\ \text{“ ”},\\ \mathit{ToLower}(\mathit{SubString}(v, \mathit{WordToken}, 2)))$$

End-User Programming - FlashExtract


Ana Trujillo
357 21th Place SE
Redmond, WA
(757) 555-1634

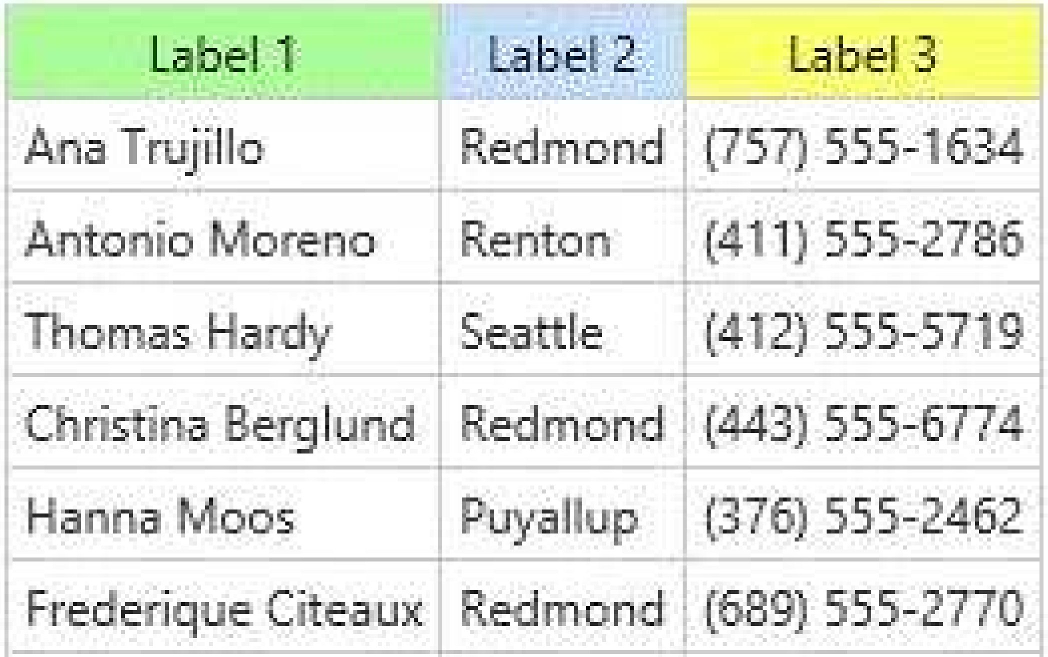
Antonio Moreno
515 93th Lane
Renton, WA
(411) 555-2786

Thomas Hardy
742 17th Street NE
Seattle, WA
(412) 555-5719

Christina Berglund
475 22th Lane
Redmond, WA
(443) 555-6774

Hanna Moos
785 45th Street NE
Puyallup, WA
(376) 555-2462

Frederique Citeaux
308 66th Place
Redmond, WA
(689) 555-2770



Label 1	Label 2	Label 3
Ana Trujillo	Redmond	(757) 555-1634
Antonio Moreno	Renton	(411) 555-2786
Thomas Hardy	Seattle	(412) 555-5719
Christina Berglund	Redmond	(443) 555-6774
Hanna Moos	Puyallup	(376) 555-2462
Frederique Citeaux	Redmond	(689) 555-2770

User Highlights	**Induced program extracts fields from Database of unstructured text**

Inductive Programming

- Earliest work in 1970s (Plotkin, 1971, Summers, 1975)
- Recent strong revival of interest, both academia and industry
- Inter-disciplinary research area
- Computer Science, Artificial Intelligence and Cognitive Science
- Automatic synthesis of programs from examples
- Inductive Functional Programming
- Inductive Logic Programming

Inductive Functional Programming

- Induction from Examples of Functional Programming
- Functional Programming Framework, deterministic
- Background Knowledge B - set of functions
- Examples E - set of ground equalities, eg factorial(5) $= 120$
- Hypothesis H - a function

Inductive Logic Programming

- Induction from Examples of Logic Programming
- Logic Programming Framework, non-deterministic
- Background Knowledge B - set of definite clause definitions
- Examples E - set of ground facts, eg larger(jupiter,earth)
- Hypothesis H - set of definite clauses
- ILP systems find H such that $B, H \models E$

IP versus Machine Learning

	Inductive Programming	Machine Learning
Examples	Small data	Big data
Form	Relations, constructors	Tables, text
Source	Humans, software	Databases, internet
Hypotheses	Programs	Network, kernel
Search	Derivation	Gradient Descent
Comprehend	High	Low
Expressivity	High	Low
Bias	Background knowledge	Bayes' Prior
Evaluation	Diverse	Error

Inductive Programming Techniques (1)
Domain-Specific Language (DSL) synthesisers
Formal Methods/Computer Science

Systems: FlashFill, FlashExtract

1. **Problem definition.** Collect common scenarios based on user studies.
2. **DSL.** Design DSL expressive enough to capture scenarios.
3. **Inductive Synthesis.** Systematically reduce problem to sub-expressions. Generate multiple DSL programs.
4. **Ranking.** Return ranking over programs.

Inductive Programming Techniques (2)
Higher-order function induction
Programming Languages/Computer Science

Systems: Igor2, MagicHaskeller

- **Background knowledge.** Consists of first-order functions, such as "+" and higher-order function such as "map".
- **Examples.** Provided as equations, eg f [[5 , 7] , [12 , 3]] = [12 , 15] .
- **Inductive Synthesis.** Searches function space, eg MagicHaskeller gives f = (map,sum).

MagicHaskeller demo:

`http://nautilus.cs.miyazaki-u.ac.jp/~skata/MagicHaskeller.html`

Inductive Programming Techniques (3)
Meta-Interpretive Learning
Artificial Intelligence

Systems: Metagol

- **Background knowledge.** Consists of first-order predicates, such as “copyword” and meta-level predicates such as “while” and MetaRules such as “Composition”.
- **Examples.** Provided as ground facts, eg transform(“john”,“John”) .
- **Inductive Synthesis.** Searches predicate space and invents predicates, eg Metagol gives transform(X,Y) $\leftarrow$ makeupper(X,Z), copyword(Z,Y).

Metagol demo: http://metagol.doc.ic.ac.uk
Metagol code: https://github.com/metagol/metagol

Challenges:
Complexity and Compositionality

- **Large search space.** How do we reduce the size of the search space?
- **Complexity of programs.** How do we minimise the complexity of the learned program?
- **Complex tasks.** How do we decompose tasks to be learned into subtasks?

Challenges:
Domain change

- **New domain.** Developing a new application area for Inductive Programming requires a large investment of time and effort.
- **Transfer.** Can we use ideas from Transfer Learning to allow IP systems to be re-used in a new domain related to previous ones?
- **First-order re-use.** How can background functions and predicates be re-used effectively?
- **Meta-level re-use.** How can meta-level functions and predicates be re-used effectively?

Challenges:
Validation and Comprehensibility

- **Understandability.** Many invented predicates. Generate names to reflect semantics?
- **Abstractions.** Abstractions to explain programs?
- **Confidence measures.** Statistical measures to indicate areas of the program which have high empirical support?
- **Pictures.** Pictures generated to indicate what a program does?
- **Explanations.** Explanations of a program in Natural Language to help user to understand it?

Challenges:
Noise tolerance

- **Noise.** Real world data often noisy. Values missing or incorrect.
- **Representation.** Some values might occur in different formats, eg dates and numbers.
- **Background errors.** Background knowledge may contain errors.
- **ML approach.** Some existing approaches can be imported from ML literature.
- **One-shot noise.** ML does not address how noise treated for one-shot learning. Problem for IP.

Challenges:
Making IP Cognitive

- **Human interface.** IP involves interaction with human beings.
- **Few examples.** Cognitive Science shows humans learn complex ideas from small numbers of positive examples.
- **Background knowledge.** Humans learn using large amounts of background knowledge.
- **Life-Long Learning.** Humans learn continuously and incrementally.
- **Interaction.** Human-Computer interactions need to be more human-like.

Summary

- End-user programming - allow world's population to program complex tasks by example.
- Inductive Programming (IP) - emerging inter-disciplinary research area.
- ILP and IFP - IP areas representing examples/background/hypotheses as logic/ functional programs.
- Differences between IP and Machine Learning.
- Search techniques include DSL, Meta-synthesis, constraint solving, Meta-Interpretive Learning.
- Challenges - Domain change, Validation, Noise, Cognitive IP.