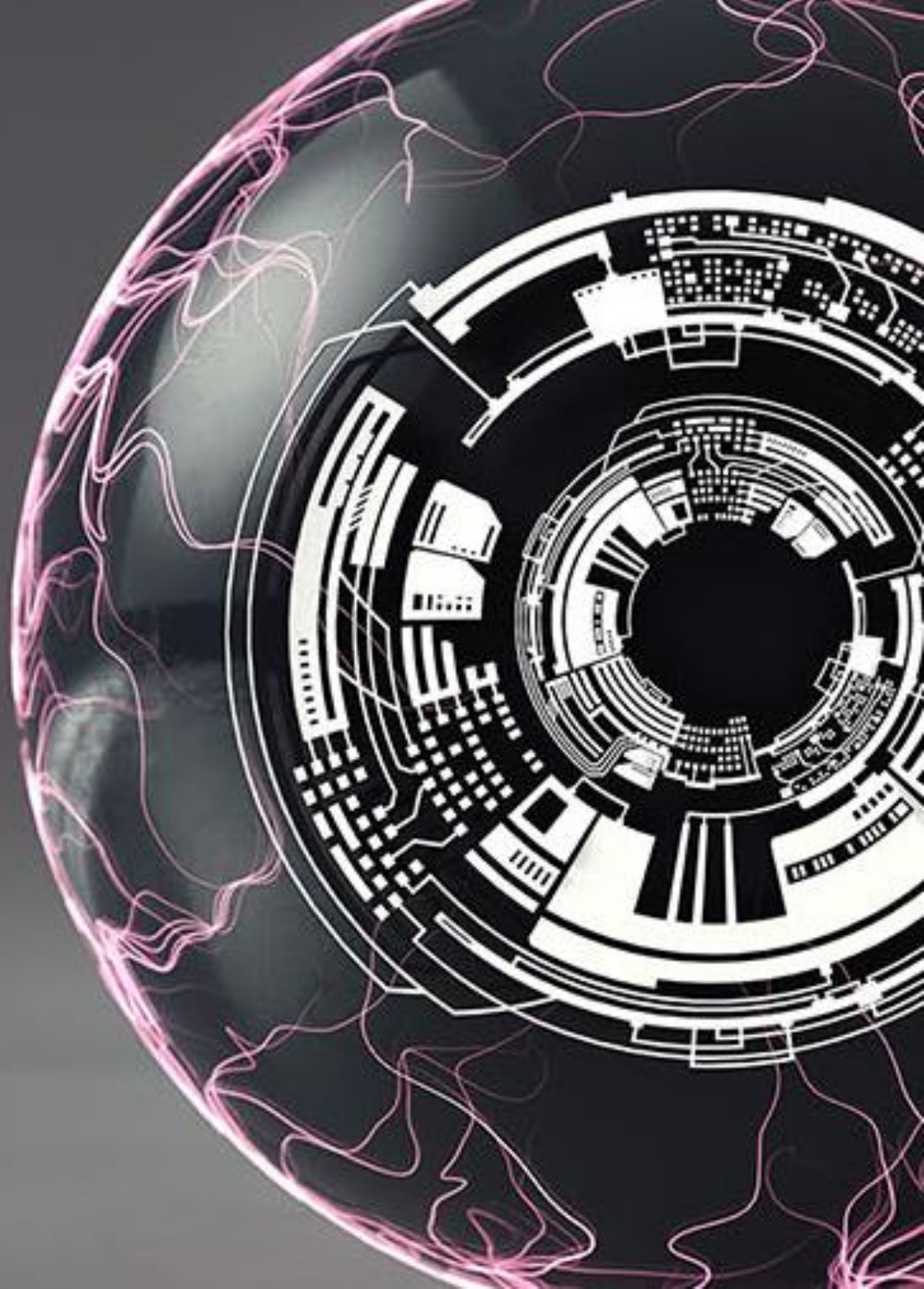

Microsoft
Microsoft Research
Faculty
Summit
2016

Microsoft

Teaching is Programming

Patrice Simard



What is Machine Teaching?

- It is about making the teacher better at creating functions *given* the learner
  - The teacher is human
  - The learner is an ML algorithm
  - ML is about making the learner better
  - Machine Teaching is about making the teacher better

Programming Assumptions

- Any computable function can be programmed (Church-Turing Thesis)
- It can be programmed and debugged in "polynomial" time in some quantity (e.g., the number of parameters? the number of states?)

Machine Teaching Questions:

- Can any (stateless) function computable by humans be taught to a computer?
- Can it be taught in polynomial time?

And the answers are:

- Yes
- Yes

Convincing you is the topic of this talk…

Teaching and Programming are Both Hard

- Learning to program a function is not easy
  - Learn a programming language's keywords -> Programming Tetris

- Learning to train a function is not easy
  - Training a dog to listen -> training a dog to roll over
  - Training a pigeon to do arithmetic -> many steps
  - Training a baby to babble -> Training a human to write books

The key is to learn to break down the problem and make it modular

In both cases, we leverage function composability

Programming and Machine Teaching Tools (let's start by breaking up the tools)

Programming		Machine Teaching
Hardware (x64, x86,…)	↔	?
Operating System	↔	?
Framework (.NET, Java…)	↔	?
Programming Languages (Fortran, Perl, Python, C, C#,...)	↔	?
Programming Expertise (design patterns…)	↔	?
Source Control	↔	?
IDEs (Eclipse, Visual Studio,…)	↔	?

Programming and Machine Teaching Tools (let's start by breaking up the tools)

Programming		Machine Teaching
• **Hardware (x64, x86,…)**	↔	• **ML Algorithms (NNs, SVMs,…)**
• **Operating System**	↔	• **Training/Sampling services**
• **Framework (.NET, Java…)**	↔	• **ML Run Time**
• **Programming Languages (Fortran, Perl, Python, C, C#,...)**	↔	• **Features, Training set ((X,Y) pairs), Schemas,…**
• **Programming Expertise (design patterns…)**	↔	• **Teaching Expertise (feature tuning, modularity, exploration)**
• **Source Control**	↔	• **Source Control**
• **IDEs (Eclipse, Visual Studio,…)**	↔	• **IDEs**

What is the Teaching Equivalent of a Programming Bug???

- Hint: it should be agnostic to the ML model

ML Errors (Traditional View)

- E : How well we could possibly do
- E(H): How well can we do in space H
- E(H,l): How well we do in space H with l data

$$E(H,l) - E = E(H,l) - E(H) + E(H) - E$$

Generalization Error = estimation error + approximation error

over training — not enough capacity

add more examples — get better model/features

Engineering View: 4 Kind of Errors (= bugs):

- Uncertainty errors -> do nothing or label as “don’t care”
- Mislabel errors -> correct label
- Ignorance errors -> add examples
- Feature blindness errors -> add, edit, or remove features

Training Process

While not diminishing return:

While there are no training errors and not diminishing return do

Find "interesting" example (pattern, label)

If ignorance error

progress is being made

Add to training set and retrain

If training error is uncertainty error then

Tag and ignore (progress)

else if training error is mislabel then

Correct label (progress)

else Fix training error

Add, edit, or remove features (progress)

Retrain

Fixing Color Blindness

The IDE can assist the teacher with “decomposability”

- IDE helps with exploration by scoring large amount of data
  - Teacher creates filters to discover new color blindness errors
- IDE suggests new features (based on training set errors)
  - Teacher select best semantic features (goal is not to fix training set)
- IDE can suggest feature refinements (using unlabeled data)
  - Teacher fine tune sub-concept

Usage

- LUIS (www.luis.ai). POCs: Jason Williams, Riham Mansour
  - Custom language understanding for short text, like queries and utterances
  - 1000s of developers including teams at Microsoft, in fortune 500 companies, and startups.
  - 1,000,000s of access point calls
- Bing Local (Entity Extraction from the Web)
- Cortana (utterances)
- Microsoft Band (email)

Demo

https://www.microsoft.com/en-us/research/video/introduction-to-luis

Conclusion

- Teaching is Programming!
- Teaching today is where programming was 30 years ago
- We can leverage 3 decades of experience with programming (and ML) to makes teaching as easy (actually much easier) than programming.
- Building a Machine Teaching IDE would change how and where ML is used and bring considerable value to the world.
- Our first prototype has enjoyed spontaneous internal and external adoption. We are overwhelmed with requests and rewriting for scale.