PART I :

MACHINE LEARNING

POPULAR SPECIFICATIONS

- **examples** : Inductive program synthesis
  ≡ Programming by Example
  - ⊕ Clear notion of a solution program
  - ⊖ Very partial specification : many solution programs
- **natural language specification** :
  - ⊕ easy to write
  - ⊖ ambiguous : no notion of a solution program

What we'll say here applies to both

FIRST IDEAS

- 1989, Solomonoff: use specification for creating statistics
- 2013, Menon et al: learn statistics from specification

DEEP LEARNING

- 2017, Balog et al: DeepCoder neural networks make predictions

LLMs

- 2021, OpenAI: "If we can translate English to French, then we can do code generation: translate language to code!"

AN EXAMPLE

Examples: Input → Output

[1,5,4,2] → [2,4]

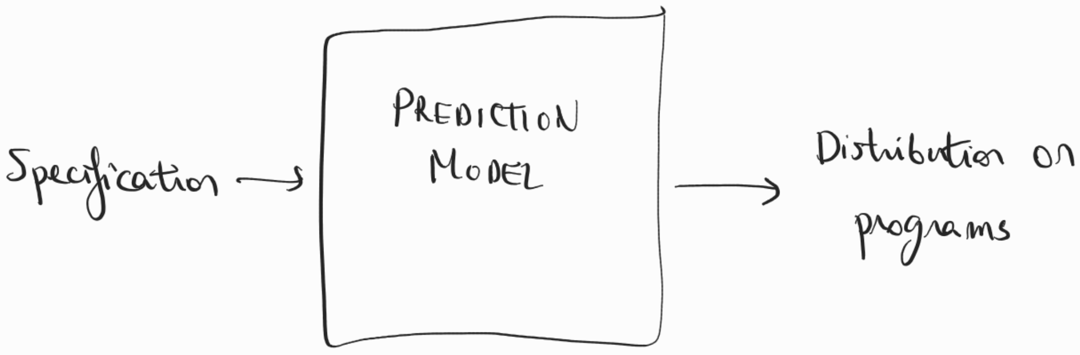
[6,3,0,8] → [0,6,8]

Key idea patterns found in specification reveal which constructs are used in a solution program

Patterns:
- all values in output are even
- outputs are sorted
- values in outputs appear in inputs

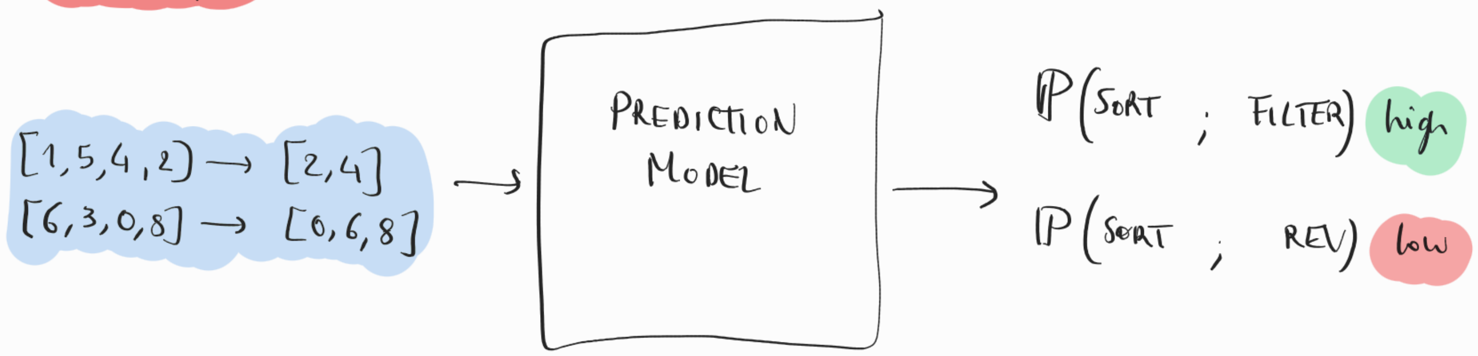
⋮

PREDICTIONS



Loss function : $\max\limits_{\text{parameters}} \ \text{Probability}(\text{Program} \models \text{Specification})$

Example :


[1,5,4,2] → [2,4]
[6,3,0,8] → [0,6,8]
PREDICTION MODEL
$\mathbb{P}$(SORT ; FILTER) high
$\mathbb{P}$(SORT ; REV) low

The prediction model induces:

$$\mathcal{D}(\text{program} \mid \text{specification})$$

the larger $\mathcal{D}(\text{Prog} \mid \text{Spec})$ the more likely Prog $\models$ Spec

TRAINING

- Find a dataset of $(\mathrm{Spec}_i, \mathrm{Prog}_i)_{i \in [1,N]}$

- Train the prediction model to maximise

$$\sum_{i=1}^{N} \mathcal{D}(\mathrm{Prog}_i \mid \mathrm{Spec}_i)$$

TWO ISSUES

- In practice hard to FIND datasets :

SYNTHETIC datasets ↓ DATA GENERATION

scrape the internet

problem : very low quality

- Program aliasing makes learning harder :

$$\begin{cases} \mathrm{Prog}_1 \models \mathrm{Spec} \\ \mathrm{Prog}_2 \models \mathrm{Spec} \end{cases} \Rightarrow \begin{array}{l} \mathcal{D}(\mathrm{prog}_1 \mid \mathrm{Spec}) = 1/2 \\ \mathcal{D}(\mathrm{prog}_2 \mid \mathrm{Spec}) = 1/2 \end{array}$$

→ "programming style" of LLMs

DATA GENERATION

Step 1: Program generation

Step 2: For each program, Specification generation

Both are non trivial:

- covering criteria? ($\not\approx$ testing)
- addressing program aliasing?

A WIDELY OPEN PROBLEM !