Introduction

What is thought? How can we describe the intelligent inferences made in everyday human reasoning and learning? How can we engineer intelligent machines? The computational theory of mind aims to answer these questions starting from the hypothesis that the mind is a computer, mental representations are computer programs, and thinking is a computational process – running a computer program.

But what kind of program? A natural assumption is that these programs take the inputs – percepts from the senses, facts from memory, etc – and compute the outputs – the intelligent behaviors. Thus the mental representations that lead to thinking are functions from inputs to outputs. However, this input-output view suffers from a combinatorial explosion: we must posit an input-output program for each task in which humans draw intelligent inferences.

A different approach is to assume that mental representations are more like theories in science: pieces of knowledge that can support many inferences in many different situations. For instance, Newton's theory of motion makes predictions about infinitely many different configurations of objects and can be used to reason both forward in time and backward, from final state of a physical system to the initial state. The *generative* approach to cognition posits that some mental representations are more like theories in this way: they capture general descriptions of how the world *works*. A generative model describes a process, usually one by which observable data is generated, that represents knowledge about the causal structure of the world. These generative processes are simplified "working models" of a domain. Other mental computations operate over these generative models to draw inferences: many different questions can be answered by interrogating the mental model. This contrasts to a more procedural or mechanistic approach in which knowledge represents the input-output mapping for a particular question directly.

It is possible to use deterministic generative models to describe possible ways a process could unfold, but due to sparsity of observations or actual randomness there will often be many ways that our observations could have been generated. How can we choose amongst them? Probability theory provides a system for reasoning under exactly this kind of uncertainty. Probabilistic generative models describe processes which unfold with some amount of randomness, and probabilistic inference describes ways to ask questions of such processes. This book is concerned with the knowledge that can be represented by probabilistic generative models and the inferences that can be drawn from them.

In order to make the idea of generative models precise we want a formal language that is designed to express the kinds of knowledge individuals have about the world. This language should be universal in the sense that it should be able to express any (computable) process. We build on the representational power of programming languages because they describe computational processes and capture the idea that what is important is causal dependence—in particular programs do not focus on the sequence of time, but rather on which events influence which other events (the "flow of data"). We introduce randomness into this language to construct a *probabilistic* programming language (PPL), and describe conditional inferences in this language. By using a PPL we are able to parsimoniously describe rich generative model structures and explore the inference patterns that emerge from the interaction of model and data.

Reading & Discussion: Readings (/readings/introduction.html)

Next chapter: 2. Generative models (/chapters/generative-models.html)