Philipp Gabler, BSc

Automatic Graph Tracking in Dynamic Probabilistic Programs via Source Transformations

Master's Thesis

to achieve the university degree of
Master of Science

submitted to
Graz University of Technology

Supervisor
Univ.-Prof. Dipl.-Ing. Dr. mont. Franz Pernkopf

Co-supervisor
Dipl.-Ing. Martin Trapp, BSc

Institute of Signal Processing and Speech Communication

Faculty of Electrical and Information Engineering

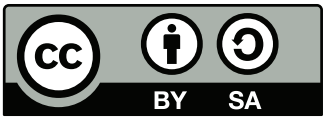
Graz, XXXX 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature



The LaTeX source of this document is available at
https://github.com/phipsgabler/master-thesis
or upon request from the author.[1]

[1] pgabler@student.tugraz.at

Abstract

This thesis proposes a novel approach for the implementation of a tracking system to facilitate program analysis, based on program transformations. The system is then applied to a specific problem in the field of probabilistic programming.

The main contribution is a general system for the extraction of rich computation graphs in the Julia programming language, based on a transformation of the intermediate representation (IR) used by the compiler. These graphs contain the whole recursive structure of any Julia program in terms of executed IR instructions. The system is flexible enough to be used for multiple purposes that require dynamic program analysis or abstract interpretation, such as automatic differentiation or dependency analysis.

The second part of the thesis describes the application of this graph tracking system to probabilistic programs written for Turing, a probabilistic programming system implemented as an embedded language within Julia. Through this, an executed Turing model can be analyzed, and the dependency structure of involved random variables be extracted from it. Given this structure, analytical Gibbs conditionals can be calculated and passed to Turing's inference mechanism, where they are used in Markov-Chain Monte Carlo samplers approximating the modelled distribution.

Contents

Notation

$\mathbb{P}[\Theta \in A \mid X = x]$	Random variables and their realizations will usually be denoted with upper and lower case letters, respectively; sets with uppercase letters.
$\mathbb{E}[X]$, $\mathbb{V}_X[Y]$	Expectation and variance; if necessary, the variable with respect to which the moment is taken is indicated.
$f(x)$, $\phi_Z(x)$	Density function are named using letters commonly used for functions, with an optional subscript indicating the random variable they belong to.
$p(x, y \mid z)$	The usual abuse of notation with the letter "p" standing for any density indicated by the names of the variables given to it is used as well (in this case, $f_{X,Y\mid Z}$ is implied).
$\int p(x)\,\mathrm{d}x = 1$	Integrals over the whole domain of a density are written as indefinite integrals, where the usage is clear.

1 Introduction

1.1 Problem Description

1.2 Related Work

2 Background

This section provides the background for the concepts used later in chapters 3 and 4. On the one hand, it gives a quick overview of Baysian inference and probabilistic programming in general, necessary to understand the requirements and usual approaches of probabilitic programming systems.

On the other hand, the machinery and language used to develop the graph tracking system forming the main part of the work are described. This consists firstly of a short introduction to graph tracking and source-to-source automatic differentiation, which contain many ideas and terminology that will be used later, and often provided inspiration. Second, the basic notions and techniques of the Julia compilation process, as well as the language's metaprogramming capabilities are described, which form the basis of the implementation.

2.1 Bayesian Inference and Probabilistic Programming

Probabilistic modelling is a technique for modelling phenomena based on the assumption that observables can be fully described through some stochastic process. When we assume this process to belong to a certain family of processes, the estimation of the "best" process is a form of learning: if we have a good description of how obserations are generated, we can make summary statements about the whole population (descriptive statistics) or make predictions about new observations. When observations come in pairs

Within a Baysian statistical framework, we assume that the family of processes that is used

bla[1]

$$p(\theta \mid x) = \frac{p(x \mid \theta)\, p(\Theta)}{\int p(x \mid t)\, \mathrm{d}t} \tag{2.1}$$

This kind of learning is called Bayesian inference.

When the distributions involved form a sufficiently "nice" combination, e.g., when they form a conjugate pair, the posterior distribution can be calculated analytically. In general, however, this

[1]Note the abuse of notation with $p()$ and the integral; see .

2.1.1 Probabilistic Programming

Probabilistic programming is a means of describing probabilistic models through the syntax of a programming language. Probabilistic programms distinguish themselves from normal programs by the possibility of being sampled from conditionally, with some of the internal variables fixed to observed values. While probabilistic programming systems are often implemented as separate, domain-specific languages, they can also be embedded into "host" programming languages with sufficient syntactic flexibility. The latter is advantageous if one wants to use regular general-purpose programming constructs or interact with other functionalities of the host language.

2.2 Computation Graphs and Automatic Differentiation

2.3 Metaprogramming and Compilation in Julia

3 Implementation of Dynamic Graph Tracking in Julia

3.1 Automatic Graph Tracking and Extended Wengert Lists

4 Graph Tracking in Probabilistic Models

5 Discussion

5.1 Future Work

Colophon

This document was typeset using the pdfLaTeX typesetting system, with the memoir document class. The body text is set in 11 pt Linux Libertine, enhanced by the microtype package. Other fonts include Biolinum and Inconsolata.

The document source has been written in Emacs with AUCTeX mode, using TeXworks as PDF viewer.