Automating Expectation Maximixation

Robert Zinkov
Department of Engineering Science
University of Oxford
zinkov@robots.ox.ac.uk

Abstract

Existing probabilistic programming systems frequently treat machine learning problems as purely inference tasks. But real-world problems often require a novel combination of inference and optimization. For example when implementing Expectation Maximization for a model, one must alternate between performing an inference and an optimization step. We show that by treating optimization as a program transformation it is possible to implement Expectation Maximization in a generic and composable way.

1 Introduction

Solving modern machine learning problems requires a hybrid and compositional approach. Many modern algorithms are a mixture of inference and optimization steps. For example, stochastic variational inference [Kucukelbir et al., 2015] requires us to both take gradients and sample from a probability distribution. Unfortunately, existing machine learning frameworks make it difficult to describe these algorithms. The inference and optimization steps have to be kept segregated, and can't be easily combined in many of these existing systems.

In this paper, we propose representing inference and optimization as program transformations in the spirit of Zinkov and Shan [2017], Ścibior and Ghahramani [2016]. The main contribution of this work is to show that Expectation Maximization [Dempster et al., 1977] can be implemented in a generic and composable way. Further this composition allows us to apply compiler optimizations to both the inference and optimization steps. This allows us to use closed-form analytic solutions and exact inference where possible in problems without requiring that knowledge from a user.

2 Related Work

Although systems like Stan [Carpenter et al., 2017] and PyMC [Salvatier et al., 2016] offer the ability to compute the maximum a posterior (MAP) of a model, they lack the ability to interleave this computation with any inference steps. This makes them unusable for models with any latent variables. Rainforth et al. [2016] were the first to show how to perform marginal maximum a posteriori inference in a probabilistic programming system. Our approach most mimics that of Edward [Tran et al., 2016] in that we also represent optimization and inference symbolically. We go further than Edward by then directly optimizing this symbolic expression to obtain closed-form analytic solutions to our problems.

3 Approach

We represent both optimization and inference steps as programs. This allows us to optimize these programs and replace them with closed-form solutions if they can be statically discovered by the compiler.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

```
EMupdate(f :: θ → p(x, z), x) =
  fn θ:
    pZ = condition(f(θ), x) # obtain p(z | x, θ)
    Q = fn θ′: expect(pZ,                              #
                      fn z: density(f(θ′),(x, z))) # E-step
    argmax(Q) # M-step
```

Figure 1: Probabilistic programming code for EM

We illustrate the approach by considering how we might represent computing the maximum a posteriori (MAP) of a Gaussian distribution with unknown mean θ.

$$x_i \sim \mathcal{N}(\theta, 1), \qquad i = 1, 2, \ldots n \tag{1}$$

We can pose this as asking for the argmax of the density of the distribution with respect to θ.

$$\underset{\theta}{\text{argmax}} \exp\left(\sum_i (x_i - \theta)^2\right) \tag{2}$$

Which has a closed-form solution as $\frac{\sum_i x_i}{n}$. This closed-form solution while often found by hand can also be found symbolically using a computer algebra system.

In our system, we represent Expectation Maximixation (EM) using four program transformations `expect`, `condition`, `density` and `argmax`. The `expect`, `condition` and `density` program transformations have the exact same implementation as used in Zinkov and Shan [2017]. The `argmax` program transformation consists of sending the expression to a computer algebra system, where the expression is differentiated and an algebra solver is used to find an exact solution.

The EM transformation itself can then be defined as shown in Figure 1. We take as input a function from the parameters to a joint probability distribution of the observed and latent random variables $(\mathbf{x}, \mathbf{z})$. We then return a symbolic expression representing a function from the old parameters θ to the new parameters θ'. The program transformation can then be combined with some boilerplate code for checking convergence and initializing the parameters. Nothing in this implementation is model-specific and may be applied to any probabilistic model.

4 Example

We demonstrate the power of our approach on a preliminary example of applying EM to a Gaussian mixture model. In a Gaussian mixture model, we are given a set of points $\mathbf{x}$ that we know belong to K clusters, and we trying to learn the centers of these clusters μ_k and the mixture proportions π_k.

```
fn (mu, pi):
  z <~ for i of range(n): categorical(pi)
  x <~ for i of range(n): normal(mu[z[i]], 1)
  return (x, z)
```

We could pass our program as the first argument to `EMupdate`, and our data as the second argument.

5 Discussion

In this paper, we demonstrate that probabilistic programming systems can readily integrate algorithms that mix optimization and inference steps. These algorithms can be implemented independently. The approach can also be extended by changing the call to expectation with something which samples from the distribution, in this way we could obtain the Monte Carlo EM Wei and Tanner [1990] with relatively little work.

References

Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32, 2017. ISSN 1548-7660. doi: 10.18637/jss.v076.i01. URL `https://www.jstatsoft.org/v076/i01`.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic variational inference in stan. In *Advances in neural information processing systems*, pages 568–576, 2015.

Tom Rainforth, Tuan Anh Le, Jan-Willem van de Meent, Michael A Osborne, and Frank Wood. Bayesian Optimization for Probabilistic Programs. In *Advances in Neural Information Processing Systems*, pages 280–288, 2016.

John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

Adam Ścibior and Zoubin Ghahramani. Modular construction of Bayesian inference algorithms. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

Greg CG Wei and Martin A Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. *Journal of the American statistical Association*, 85(411): 699–704, 1990.

Robert Zinkov and Chung-chieh Shan. Composing inference algorithms as program transformations. In *Proceedings of Uncertainty in Artificial Intelligence*, 2017.