Swamps of Numerical Integration
Monte Carlo Heights
Normalizing Flow
Control Line Hills Of Time
Backprop Bay

Swamps of Numerical Integration
Monte Carlo Heights
Normalizing Flow
Forward Backward Beach
Quicksand Bay
Backprop Bay

Forward backward algorithm

Cheng Soon Ong
Marc Peter Deisenroth

December 2020

Motivation

Key idea

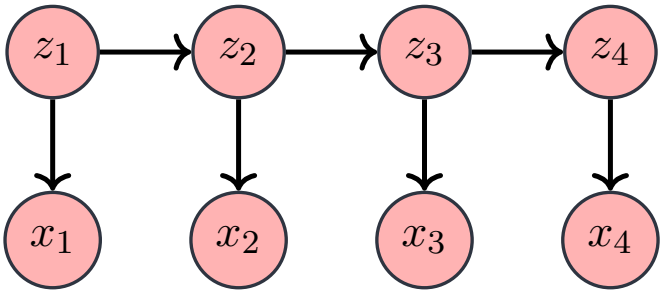
Use automatic differentiation to derive the forward backward algorithm.

Related algorithms:

- dynamic programming
- message passing
- belief propagation
- inside outside

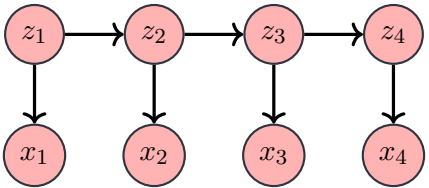
Hidden Markov model

Consider a hidden Markov model with 4 hidden nodes and 4 observed nodes.



Other names: Directed graphical model, Bayesian network

Hidden Markov model



Probabilistic inference questions:

- Generate new observations x for hidden states z
- Estimate most likely hidden states z given observations x
- Compute marginal distribution $p(z_3)$
- Learn the parameters, given observations x

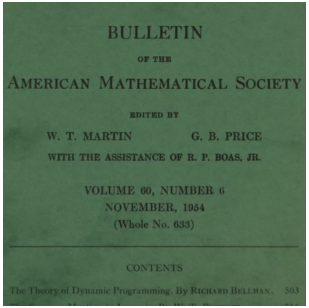
Compute posterior marginals

The forward–backward algorithm

calculates for all hidden state variables $z_1, \ldots, z_4$ the distribution $p(z_t|x_1, \ldots, x_4)$.

Pointers to literature

- Hidden Markov model (Rabiner, 1989; Cappé et al., 2006)
- Dynamic programming, Bellman equations (Sutton and Barto, 1998; Bertsekas, 2018)
- Automatic differentiation applied to arithmetic circuits
  (Darwiche, 2003; Brandherm and Jameson, 2004; Eisner, 2016)
- Message passing, belief propagation
  `https://tminka.github.io/papers/acml12019/`


BULLETIN
OF THE
AMERICAN MATHEMATICAL SOCIETY
EDITED BY
W. T. MARTIN G. B. PRICE
WITH THE ASSISTANCE OF R. P. BOAS, JR.
VOLUME 60, NUMBER 6
NOVEMBER, 1954
(Whole No. 633)
CONTENTS
The Theory of Dynamic Programming. By RICHARD BELLMAN. 503

Recall: Automatic differentiation

Think of the chain rule of differentiation, **but**

Cache intermediate results

It turns out that we can reduce computational cost of computing the gradient if we cache intermediate results.

Trade off computational complexity for space complexity.

Automatic differentiation: $+$ and $\times$

We consider the special case of arithmetic circuits which only consists of multiplies (products) and adds (sums).

Key idea

Observe that the gradient of multiply scales the value of the other input

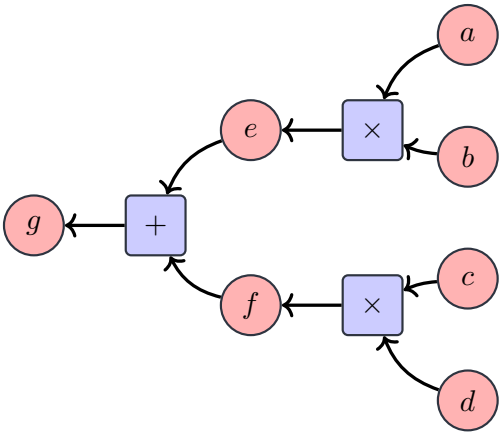
By considering an equation of the form

$$e = ab\,,$$

we observe that

$$\frac{\partial e}{\partial a} = b \quad \text{and} \quad \frac{\partial e}{\partial b} = a\,.$$

Multiply then add



Read from right to left.

Multiply then add

- Consider the gradient of g with respect to a and b
- By the chain rule,

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial e}\frac{\partial e}{\partial a} \quad \text{and} \quad \frac{\partial g}{\partial b} = \frac{\partial g}{\partial e}\frac{\partial e}{\partial b}.$$

Multiply then add

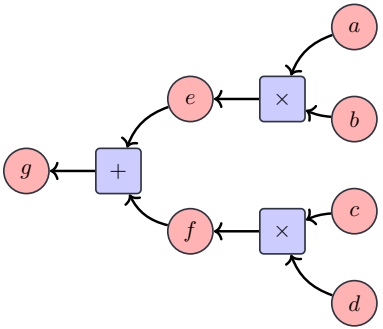
- Consider the gradient of g with respect to a and b
- By the chain rule,

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial e}\frac{\partial e}{\partial a} \quad \text{and} \quad \frac{\partial g}{\partial b} = \frac{\partial g}{\partial e}\frac{\partial e}{\partial b}.$$

- Substituting the partial differential for the product ($e = ab$)

$$\frac{\partial g}{\partial a} = \frac{\partial g}{\partial e}b \quad \text{and} \quad \frac{\partial g}{\partial b} = a\frac{\partial g}{\partial e}.$$

- The key variable is **the middle variable** e



Consider a pair $(a, \overleftarrow{a})$

Key idea

Think of partial differentiation with respect to an intemediate variable as an function

- Think of $\frac{\partial}{\partial e}$ as an operation on the variable g

$$\frac{\partial}{\partial a}(g) = \frac{\partial}{\partial e}(g) \cdot b \quad \text{and} \quad \frac{\partial}{\partial b}(g) = a \cdot \frac{\partial}{\partial e}(g)$$

- automatic differentiation augments each variable (for example a) with an **adjoint** variable $\overleftarrow{a}$ to form an adjoint pair $(a, \overleftarrow{a})$
- The adjoint $\overleftarrow{a}$ of a is the function that takes the partial differential $\frac{\partial}{\partial a}$

Dual numbers

Key idea

The adjoint $\overleftarrow{a}$ of a is the partial differential function $\frac{\partial}{\partial a}$.

Given $e = ab$,

we obtain a pair of adjoint equations

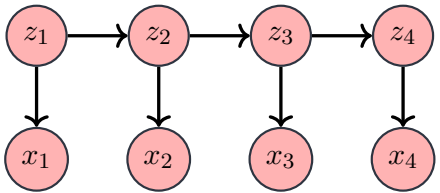
$$\overleftarrow{a} = \overleftarrow{e}\, b \quad \text{and}$$
$$\overleftarrow{b} = a\, \overleftarrow{e}$$

- The gradient of multiply scales the value of the other input
- Generalizes to a product of n variables, generating n corresponding adjoint equations.

Back to hidden Markov models

Key idea

Derive the backward pass by considering the trellis.
Obtain the forward pass by reverse mode automatic differentiation.



Focus on the recursive part of the flow of information from right to left, in particular from node z_3 to z_2

Backward algorithm

- Focus on the recursive part of the flow of information from node z_3 to z_2
- Assume that each hidden node z_t has two possible states $\{A, B\}$
- Denote the message originating at z_3, going left, as the vector $\boldsymbol{\beta}(z_3)$

$$\boldsymbol{\beta}(z_3) = \begin{bmatrix} \beta(z_3^A) \\ \beta(z_3^B) \end{bmatrix}$$

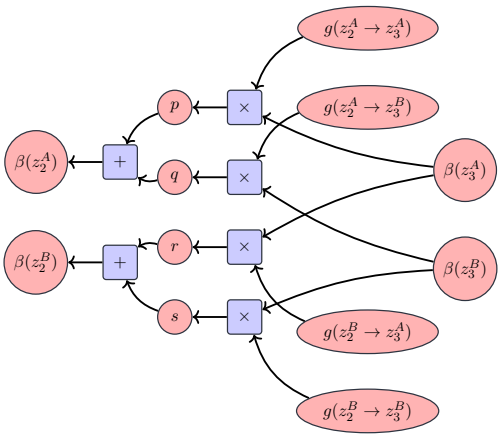
Backward algorithm

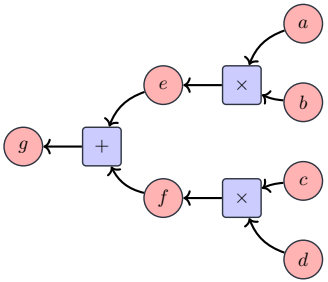
- To compute the corresponding message $\boldsymbol{\beta}(z_2)$ we have the following recursion

$$\boldsymbol{\beta}(z_2^A) = g(z_2^A \to z_3^A)\boldsymbol{\beta}(z_3^A) + g(z_2^A \to z_3^B)\boldsymbol{\beta}(z_3^B)$$
$$\boldsymbol{\beta}(z_2^B) = g(z_2^B \to z_3^A)\boldsymbol{\beta}(z_3^A) + g(z_2^B \to z_3^B)\boldsymbol{\beta}(z_3^B)\,.$$

- This transition also includes the emission of the symbol x_2.
- We have introduced the functions $g(z_2^A \to z_3^B)$ to denote the transition from z_2^A to z_3^B.

Computational graph





Adjoint patterns

Key idea

Perform reverse mode automatic differentiation on multiply and add network

Algorithm 7 The forward-backward algorithm

procedure FORWARD-BACKWARD($\mathcal{G}$, $\mathbf{w}$)
  $Z :=$ BACKWARD($\mathcal{G}$, $\mathbf{w}$) ▷ *also sets* $\beta[\cdots]$
  initialize all $\alpha[\cdots]$ to 0
  $\alpha[\text{ROOT}^0]$ += 1 ▷ *sets* $\eth Z = 1$
  for $A \in \mathcal{N}$: ▷ *starting rules*
    $\alpha[\text{ROOT} \to A]$ += $\alpha[\text{ROOT}^0]\ \beta[A^1]$
    $\alpha[A^1]$ += $\alpha[\text{ROOT}^0]\ \mathcal{G}(\text{ROOT} \to A)$
  for $j := 1$ to $n - 1$:
    for $A, B \in \mathcal{N}$: ▷ *transition rules*
      $\alpha[A \to w_j\ B]$ += $\alpha[A^j]\ \beta[B^{j+1}]$
      $\alpha[B^{j+1}]$ += $\alpha[A^j]\ \mathcal{G}(A \to w_j\ B)$
  for $A \in \mathcal{N}$: ▷ *stopping rules*
    $\mathcal{G}(A \to w_n)$ += $\alpha[A^n]$
  for $R \in \mathcal{R}$: ▷ *expected rule counts*
    $c(R) := \alpha[R] \cdot \mathcal{G}(R)/Z$

Eisner (2016)

Can use for more general functions

Intuition

Use chain rule to linearize function, and reverse mode autodiff to accumulate

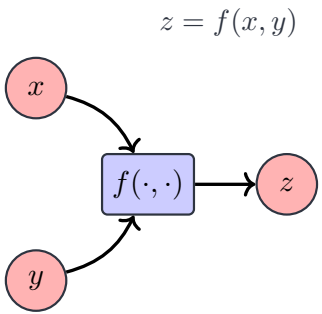
forward pass by the chain rule

$$\frac{dz}{dx} = \frac{\partial z}{\partial x}\partial x + \frac{\partial z}{\partial y}\partial y$$

reverse pass

$$\overleftarrow{x} += \overleftarrow{z} \times \frac{\partial z}{\partial x}$$

$$\overleftarrow{y} += \overleftarrow{z} \times \frac{\partial z}{\partial y}$$



Summary

- Consider the special case of $+$ and $\times$ computation
- Hidden Markov models are a special case of dynamic programming
- Given the backward algorithm, can derive the forward algorithm by reverse mode automatic differentiation

Backpropagation is just...

simplified message passing in probabilistic graphical models

References

Bellman, R. E. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60:503–516.

Bertsekas, D. P. (2018). Abstract dynamic programming.

Brandherm, B. and Jameson, A. (2004). An extension of the differential approach for bayesian network inference to dynamic bayesian networks. *International Journal of Intelligent Systems*, 19(8):727–748.

Cappé, O., Moulines, E., and Rydén, T. (2006). *Inference in hidden Markov models*. Springer Science & Business Media.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280305.

Eisner, J. M. (2016). Inside-outside and forward-backward algorithms are just backprop. In *Structured Prediction Workshop at EMNLP*.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.