Belief propagation on strings

Boris Yangel

Last modified: November 27, 2013

Abstract

This document describes a way of performing belief propagation in models with string variables, with factors corresponding to widely used operations on strings such as `Substring`, `Contains`, `Replace`, `Concat` etc.

Contents

1 Notation

We denote automata by capital letters A, B etc or, in case of a specific automaton, by a string starting from a capital letter, such as Unif. However, when an automaton represents a BP message, we denote it by μ with a subscript describing the source and the destination of the message, like $\mu_{f \to s}$. Notation for transducers is the same, except that we use a different font: $\mathcal{T}$, $\mathcal{U}$, $\mathcal{Copy}$. Lowercase letters and sequences of lowercase letters denote variables: s, t, str.

The notation $s \cdot t$ stands for the concatenation of strings s and t.

2 Automata

We represent marginals and messages from and to string variables by weighted finite state automata. Weighted finite state automaton, which we'll refer to simply as automaton from now on, is a function mapping strings to real values. Not every such function is a finite state automaton though. For an explaination of what a finite state automaton really is and what are the limitations of this class of functions we refer the reader to [**?**], while this paper takes a different approach. We instead define a set of atomic automata, a number of operations w.r.t. which automata are closed, and then simply use all the automata that can be obtained from the atomic ones using the proposed operations.

2.1 Atomic automata

Name	Notation	Definition
Uniform of characters	Unif_C	Maps every string consisting entirely of characters from the set C to 1 and all other strings to 0.
Single character	Char_C	Maps every string consisting of a single character from the set C to 1 and all other strings to 0.

If the set C consists of all possible characters, we denote Unif_C simply by Unif, and Char_C by Char.

2.2 Operations on automata

Name	Notation	Definition
Scale	$\alpha\mathrm{A}$	$(\alpha\mathrm{A})(s) \coloneqq \alpha\mathrm{A}(s)$
Sum	$\mathrm{A}+\mathrm{B}$	$(\mathrm{A}+\mathrm{B})(s) \coloneqq \mathrm{A}(s)+\mathrm{B}(s)$
Product	AB	$(\mathrm{AB})(s) \coloneqq \mathrm{A}(s)\mathrm{B}(s)$
Concatenation	$\mathrm{A}\cdot\mathrm{B}$	$(\mathrm{A}\cdot\mathrm{B})(s) \coloneqq \sum_{\substack{s_1,s_2 \\ s_1\cdot s_2=s}} \mathrm{A}(s_1)\mathrm{B}(s_2)$.
Normalizer	$\mathcal{Z}(\mathrm{A})$	$\mathcal{Z}(\mathrm{A}) \coloneqq \sum_s \mathrm{A}(s)$ (defined only when the corresponding sum converges).

We denote $\mathrm{A}+(-1)\mathrm{B}$ simply by $\mathrm{A}-\mathrm{B}$.

2.3 Examples of compound automata

Notation	Definition	Value
OfLen_N	$\underbrace{\mathrm{Char}\cdot\mathrm{Char}\cdot\ldots\cdot\mathrm{Char}}_{N \text{ times}}$	1 on any string of length N, zero on everything else.

3 Transducers

In order to compute messages from a factor to a variable, we also need to employ weighted finite state transducers, which are, essentially, functions mapping pairs of sequences to real values. Following the section 2, we define the set of atomic transducers and a number of operations that allows us to produce all the transducers of interest. For a more general introduction, we refer the reader to [**?**].

3.1 Atomic transducers

Name	Notation	Definition
Copy	$\mathcal{Copy}$	$\mathcal{Copy}(s,t) := \begin{cases} 1, & s = t \\ 0 & \text{otherwise} \end{cases};$
Copy char	$\mathcal{CopyChar}$	$\mathcal{CopyChar}(s,t) := \begin{cases} 1, & s = t, \lvert s\rvert = \lvert t\rvert = 1 \\ 0 & \text{otherwise} \end{cases};$
Consume an automaton	$\mathcal{Consume}_{\mathrm{A}}$	$\mathcal{Consume}_A(s,t) := \begin{cases} \mathrm{A}(s), & t = \text{``''} \\ 0 & \text{otherwise} \end{cases};$
Produce an automaton	$\mathcal{Produce}_{\mathrm{A}}$	$\mathcal{Produce}_A(s,t) := \begin{cases} \mathrm{A}(t), & s = \text{``''} \\ 0 & \text{otherwise} \end{cases}.$

3.2 Operations on transducers

Name	Notation	Definition
Concatenation	$\mathcal{A} \cdot \mathcal{B}$	$(\mathcal{A} \cdot \mathcal{B})(s,t) := \sum_{\substack{s_1,t_1,s_2,t_2 \\ s_1 \cdot s_2 = s, t_1 \cdot t_2 = t}} \mathcal{A}(s_1,t_1)\mathcal{B}(s_2,t_2)$
Transpose	$\mathcal{A}^T$	$(\mathcal{A}^T)(s,t) := \mathcal{A}(t,s)$
Superposition	$\mathcal{B}(\mathcal{A})$	$(\mathcal{B}(\mathcal{A}))(s,t) := \sum_u A(s,u)B(u,t)$

3.3 Projection of an automaton onto a transducer

Additionaly, an automaton can be projected onto a transducer. The result of the projection is another automaton, which is defined as

$$\mathrm{proj}[\mathrm{A}, \mathcal{T}](t) := \sum_s \mathrm{A}(s)\mathcal{T}(s,t). \tag{1}$$

This operation can be very handy when computing BP messages.

3.4 Some properties of operations on transducers

Property 1.

$$\mathrm{proj}[\mathrm{A}, \mathcal{U}(\mathcal{T})] = \mathrm{proj}[\mathrm{proj}[\mathrm{A}, \mathcal{T}], \mathcal{U}]. \tag{2}$$

Proof.

$$\begin{aligned} \mathrm{proj}[\mathrm{A}, \mathcal{U}(\mathcal{T})](s,t) &= \sum_s \mathrm{A}(s) \sum_u \mathcal{T}(s,u)\mathcal{U}(u,t) = \sum_u \mathcal{U}(u,t) \sum_s \mathrm{A}(s)\mathcal{T}(s,u) = \\ &= \sum_u \mathcal{U}(u,t)\,\mathrm{proj}[\mathrm{A}, \mathcal{T}](u) = \mathrm{proj}[\mathrm{proj}[\mathrm{A}, \mathcal{T}], \mathcal{U}]. \end{aligned} \tag{3}$$

Property 2.

$$\mathcal{Z}(\mathrm{A}) = \mathrm{proj}[\mathrm{Unif}, \mathcal{Consume}_{\mathrm{A}} \cdot \mathcal{Produce}_{\mathrm{Unif}}](t) \quad \forall t. \tag{4}$$

Proof.

$$\forall t: \quad \mathcal{Z}(\mathrm{A}) = \sum_s \mathrm{A}(s) = \sum_s \mathcal{Unif}(s)\tilde{\mathrm{A}}(s,t), \tag{5}$$

where

$$\tilde{\mathrm{A}}(s,t) = \mathrm{A}(s) = \mathcal{Consume}_{\mathrm{A}} \cdot \mathcal{Produce}_{\mathrm{Unif}}(s,t). \tag{6}$$

3.5 Examples of compound transducers

Here we present a number of compound transducers that are useful for message computation.

Notation	Definition	Value
$\mathcal{Occurrences}(s, str)$	$\mathcal{Produce}_{\mathrm{Unif}} \cdot \mathcal{Copy} \cdot \mathcal{Produce}_{\mathrm{Unif}}$	The number of occurences of s in str.
$\mathcal{OccursAt}_p(s, str)$	$\mathcal{Produce}_{\mathrm{OfLen}_p} \cdot \mathcal{Copy} \cdot \mathcal{Produce}_{\mathrm{Unif}}$	1 if s occurs in str at the position p, 0 otherwise.
$\mathcal{CopyOfLen}_N(s_1, s_2)$	$\underbrace{\mathcal{CopyChar} \cdot \mathcal{CopyChar} \cdot \ldots \cdot \mathcal{CopyChar}}_{N \text{ times}}$	1 if $s_1 = s_2$ and both strings are of length N.
$\mathcal{IsSubstring}_{p,l}(s, str)$	$\mathcal{Produce}_{\mathrm{OfLen}_p} \cdot \mathcal{CopyOfLen}_l \cdot \mathcal{Produce}_{\mathrm{Unif}}$	1 if s occurs in str from the position p to $p+l$, 0 otherwise.

4 Message operators

4.1 Substring

4.1.1 Factor

$$f(str, s, p, l) = \mathbb{1}[str \text{ contains } s \text{ from } p \text{ to } p+l]. \tag{7}$$

We assume that the position p and the length l are observed.

4.1.2 Message to str

$$\begin{aligned}\mu_{f\to str}(str) &= \sum_s \mu_{s\to f}(s)\mathbb{1}[str \text{ contains } s \text{ from } p \text{ to } p+l] = \\ &= \mathrm{proj}[\mu_{s\to f}, \mathcal{IsSubstring}_{p,l}]. \end{aligned} \tag{8}$$

4.1.3 Message to s

$$\begin{aligned}\mu_{f\to s}(s) &= \sum_{str} \mu_{str\to f}(s)\mathbb{1}[str \text{ contains } s \text{ from } p \text{ to } p+l] = \\ &= \mathrm{proj}[\mu_{str\to f}, \mathcal{IsSubstring}^T_{p,l}]. \end{aligned} \tag{9}$$

4.2 Concat

4.2.1 Factor

$$f(s, s_1, s_2) = \mathbb{1}[s = s_1 \cdot s_2]. \tag{10}$$

4.2.2 Message to s

$$\mu_{f\to s}(s) = \sum_{s_1} \mu_{s_1\to f}(s_1) \sum_{s_2} \mu_{s_2\to f}(s_2) \mathbb{1}[[s = s_1 \cdot s_2] = \mu_{s_1\to f} \cdot \mu_{s_2\to f}. \tag{11}$$

4.2.3 Message to s_1

$$\begin{aligned} \mu_{f\to s_1}(s_1) = &\sum_{s} \mu_{s\to f}(s) \sum_{s_2} \mu_{s_2\to f}(s_2) \mathbb{1}[s = s_1 \cdot s_2] = \\ &= \operatorname{proj}[\mu_{s\to f}, \mathit{Copy} \cdot \mathit{Consume}_{\mu_{s_2\to f}}]. \end{aligned} \tag{12}$$

4.2.4 Message to s_2

$$\begin{aligned} \mu_{f\to s_2}(s_2) = &\sum_{s} \mu_{s\to f}(s) \sum_{s_1} \mu_{s_1\to f}(s_1) \mathbb{1}[s = s_1 \cdot s_2] = \\ &= \operatorname{proj}[\mu_{s\to f}, \mathit{Consume}_{\mu_{s_1\to f}} \cdot \mathit{Copy}]. \end{aligned} \tag{13}$$

4.3 ContainsAt

4.3.1 Factor

$$\begin{aligned} f(str, s, p, c) = &\, c\mathbb{1}[str \text{ contains } s \text{ at pos } p] + \\ &+ (1 - c)(1 - \mathbb{1}[str \text{ contains } s \text{ at pos } p]). \end{aligned} \tag{14}$$

We assume that the position p is always observed.

4.3.2 Message to c

$$\begin{aligned} \mu_{f\to c}(c = 1) = &\sum_{str} \mu_{str\to f}(str) \sum_{s} \mu_{s\to f}(s) \mathbb{1}[str \text{ contains } s \text{ at pos } p] = \\ &= \mathcal{Z}\left(\mu_{str\to f} \operatorname{proj}[\mu_{s\to f}, \mathit{OccursAt}_p]\right), \end{aligned} \tag{15}$$

$$\begin{aligned} \mu_{f\to c}(c = 0) = &\sum_{str} \mu_{str\to f}(str) \sum_{s} \mu_{s\to f}(s)(1 - \mathbb{1}[str \text{ contains } s \text{ at pos } p]) = \\ &= \mathcal{Z}(\mu_{str\to f})\mathcal{Z}(\mu_{s\to f}) - \mathcal{Z}\left(\mu_{str\to f} \operatorname{proj}[\mu_{s\to f}, \mathit{OccursAt}_p]\right). \end{aligned} \tag{16}$$

4.3.3 Message to str

$$\mu_{f\to str}(str) = \mu_{c\to f}(c=1)\sum_{s}\mu_{s\to f}(s)\mathbb{1}[str \text{ contains } s \text{ at pos } p]+$$
$$+\mu_{c\to f}(c=0)\sum_{s}\mu_{s\to f}(s)(1-\mathbb{1}[str \text{ contains } s \text{ at pos } p]) =$$
$$= (\mu_{c\to f}(c=1) - \mu_{c\to f}(c=0))\,\mathrm{proj}[\mu_{s\to f}, \mathit{OccursAt}_p] + \mu_{c\to f}(c=0)\mathrm{Unif}. \tag{17}$$

4.3.4 Message to s

$$\mu_{f\to s}(s) = \mu_{c\to f}(c=1)\sum_{str}\mu_{str\to f}(str)\mathbb{1}[str \text{ contains } s \text{ at pos } p]+$$
$$+\mu_{c\to f}(c=0)\sum_{str}\mu_{str\to f}(str)(1-\mathbb{1}[str \text{ contains } s \text{ at pos } p]) =$$
$$= (\mu_{c\to f}(c=1) - \mu_{c\to f}(c=0))\,\mathrm{proj}[\mu_{str\to f}, \mathit{OccursAt}_p^T] + \mu_{c\to f}(c=0)\mathrm{Unif}. \tag{18}$$