Handling Ties Correctly and Efficiently in Viterbi Training Using the Viterbi Semiring

Markus Saers(✉) and Dekai Wu

Department of Computer Science and Engineering,
Human Language Technology Center,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong
{masaers,dekai}@cs.ust.hk

Abstract. The handling of ties between equiprobable derivations during Viterbi training is often glossed over in research paper, whether they are broken randomly when they occur, or on an *ad-hoc* basis decided by the algorithm or implementation, or whether all equiprobable derivations are enumerated with the counts uniformly distributed among them, is left to the readers imagination. The first hurts rarely occurring rules, which run the risk of being randomly eliminated, the second suffers from algorithmic biases, and the last is correct but potentially very inefficient. We show that it is possible to Viterbi train correctly without enumerating all equiprobable best derivations. The method is analogous to expectation maximization, given that the automatic differentiation view is chosen over the reverse value/outside probability view, as the latter calculates the wrong quantity for reestimation under the Viterbi semiring. To get the automatic differentiation to work we devise an unbiased subderivative for the max function.

Keywords: Parsing · Viterbi training · Automatic differentiation
Deductive systems · Semiring parsing

1 Introduction

Conventional wisdom has it that expectation maximization is preferable over Viterbi training for reestimating generative models because all possible configurations of the hidden variables (paths through a lattice/trees in a forest) contribute proportionally to the reestimation, but Viterbi training has the potential of being significantly faster, since only the best paths/trees are needed. Goodman [6] showed that the necessary quantities for expectation maximization (EM) can be derived automatically in the form of reverse values, a generalization of backward/outside probabilities. Eisner *et al.* [5], Li and Eisner [8], and Smith [14] showed that automatic differentiation [2] of the sentence probability with respect to the rule probabilities is equivalent to reverse values, and Eisner [4]

© Springer International Publishing AG, part of Springer Nature 2018
S. T. Klein et al. (Eds.): LATA 2018, LNCS 10792, pp. 284–295, 2018.
https://doi.org/10.1007/978-3-319-77313-1_22

pointed out that it is all essentially backpropagation as we know it from neural networks [10]. In this paper we show that the equivalence between reverse values and derivatives fails to generalize away from the probability semiring, specifically: applying the two methods to the Viterbi semiring gives very different results. We further show that the backpropagation approach is indeed as helpful to Viterbi training as it is to EM training.

Viterbi training (not to be confused with Viterbi decoding: finding the best path/tree in a lattice/forest) is similar to EM in that the rule probabilities are iteratively reestimated through counts obtained by reconstructing the hidden lattice/forest using the model itself. The difference lies in how the counts are obtained: EM lets the entire lattice/forest contribute proportionally, producing expected counts, whereas only the best path/tree is counted in Viterbi training. This makes it attractive because it opens up for more efficient algorithms to be used.

There is a hidden problem with performing Viterbi training based on Viterbi decoding, in that a decoder must return either (a) *a single* path/tree, or (b) *enumerate all* best paths/trees, which harms training of highly ambiguous models. These models can have a (potentially very) large number of best paths/trees, making (a) highly approximative, and (b) highly inefficient. It is not difficult to code around these problems, but as we show in this paper, it is possible to use automatic differentiation to do it correctly and efficiently with the exact same code as EM by substituting the probability semiring with the Viterbi semiring, and differentiating it carefully.

Although this paper is framed in terms of reestimation over parse forests or lattices, the techniques are equally valid for any model that can be described in terms of a deductive system (Sect. 2.3).

2 Background

This paper synthesizes expectation maximization and Viterbi training under deductive systems using automatic differentiation; all of which are known and established methods that we will briefly review in this section.

2.1 Expectation Maximization

Expectation maximization (EM) is a method for reestimating model parameters towards a local optimum of the marginal log likelihood of some data when part of the data is hidden and has to be reconstructed by the model [3]. Rather than optimizing the likelihood directly, the Q-function—the expectation of the log likelihood—is optimized iteratively in two steps: the **expectation** step (E) calculates the Q-function, and the **maximization** step (M) reestimates the model parameters to maximize Q:

$$\text{expectation:} \qquad Q\left(\theta|\theta^{(t)}\right) = \mathrm{E}_{Z|X,\theta^{(t)}}\left[\log L\left(\theta; X, Z\right)\right] \tag{1}$$

$$\text{maximization:} \qquad \theta^{(t+1)} = \operatorname*{argmax}_{\theta} Q\left(\theta|\theta^{(t)}\right) \tag{2}$$

where X is the observed data, Z is the hidden data, and θ is the model parameters. Instead of calculating the Q-function explicitly, which is frequently intractable, the sufficient statistics q needed for the M-step is enough:

$$\text{expectation:} \qquad q^{(t)} = \mathrm{E}_Z \left[\log P \left(X, Z | \theta^{(t)} \right) \right] \tag{3}$$

$$\text{maximization:} \qquad \theta^{(t+1)} = \mathrm{MLE} \left(q^{(t)} \right) \tag{4}$$

The sufficient statistics for context-free grammars are the fractional rule counts obtained from inside/forward and outside/backward probabilities which are calculated by integrating out the hidden structure (forest/lattice), and can be generalized as such:

$$q \left(A \to \phi^0 \cdots \phi^{R-1} \right) =$$

$$p \left(A \to \phi^0 \cdots \phi^{R-1} \right) \sum_{w_{0..T} \in \mathcal{D}} \frac{1}{\alpha \left(S_{0,T} \right)} \sum_{0 \leq i_0 < i_R \leq T} \beta \left(A_{i_0, i_R} \right) \sum_{i_0 < \cdots < i_R} \prod_{j=0}^{R-1} \alpha \left(\phi^j_{i_j, i_{j+1}} \right) \tag{5}$$

where p is the rule probability function, A is a nonterminal, ϕ^i is either a nonterminal or a terminal, and $\phi^i_{s,t}$ is that same (non-)terminal covering the span from s to t, R is the number of nonterminals and terminals on the right hand side of the rule, $w_{0..T} = w_0 w_1 \cdots w_{T-1}$ is a sentence in the data $\mathcal{D}$, α is the inside/forward probability, β is the backward/outside probability, and S is the dedicated start symbol of the grammar. Saers and Wu [11] and later Eisner [4] show that reverse values can be generalized to rules, replacing most of Eq. 5 with the rule's reverse value:

$$q(r) = p(r) \sum_{w_{0..T} \in \mathcal{D}} \frac{1}{\alpha \left(S_{0,T} \right)} \beta(r) \tag{6}$$

2.2 Viterbi Training

Viterbi training consists of iteratively counting the number of times a rule occurs in the best path/tree of a sentence over the entire training data, and performing maximum likelihood estimation on those counts. This is guaranteed to approach a local optimum of the best path probability, at least for hidden Markov models [7]. Viterbi training is attractive, as it is often faster to derive only the best path/tree than to derive the entire lattice/forest. Describing it as closely as we can to EM, we have:

$$\text{counts:} \qquad c^{(t)} = \operatorname*{argmax}_z \log P \left(X, Z = z | \theta^{(t)} \right) \tag{7}$$

$$\text{maximization:} \qquad \theta^{(t+1)} = \mathrm{MLE} \left(c^{(t)} \right) \tag{8}$$

with the only difference that rather than integrating over the hidden data, we choose the hidden data configuration z that maximizes the likelihood.

Many NLP applications contain a best decoding component that provides the path or tree to collect Viterbi counts from, but there is an inherent problem hidden in this approach: It typically chooses a single path/tree—either randomly or worse: systematically—when there are multiple equiprobable paths/trees. For correct training, the rule counts have to be distributed equally to all equiprobable best paths/trees. It is possible to enumerate all such paths/trees, but the time complexity grows exponentially with the ambiguity of the model. In this paper, we show that it is possible to leverage the lattice/forest created during the forward/inside pass to efficiently and correctly solve this problem using automatic differentiation and the Viterbi semiring provided an unbiased subderivative of the max function is used.

Table 1. Common semirings.

Name	Domain	$\oplus$	$\otimes$	$\mathbf{0}$	$\mathbf{1}$
real	$\mathbb{R}$	$+$	$\times$	0	1
Boolean	{true, false}	$\vee$	$\wedge$	false	true
tropical	$\mathbb{R} \cup \infty$	min	$+$	∞	0
max-plus ("arctic")	$\mathbb{R} \cup -\infty$	max	$+$	$-\infty$	0
probability	$[0, 1]$	$+$	$\times$	0	1
Viterbi	$[0, 1]$	max	$\times$	0	1
log probability	$[-\infty, 0]$	logadd	$+$	$-\infty$	0
negative log probability (cost)	$[0, \infty]$	logadd	$+$	∞	0
log Viterbi	$[-\infty, 0]$	max	$+$	$-\infty$	0
negative log Viterbi (min cost)	$[0, \infty]$	min	$+$	∞	0

2.3 Semiring Parsing and Deductive Systems

A deductive system [6,9,12] is a way to specify how a parser uses grammar rules to construct larger constituents from smaller constituents and the input sentence. A **conclusion** b may be reached iff all the I **conditions** $a_0, a_1, \cdots, a_{I-1}$ are met. This is written as an **inference rule**:

$$\frac{a_0, a_1, \cdots, a_{I-1}}{b}$$

Conclusions are **items** (partial results such as labeled spans), and conditions are either other conclusions or **axioms** (grammar rules).

Semirings are best understood as generalizations of addition and multiplication. Formally they are tuples $(\mathcal{A}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $\mathcal{A}$ is the domain, $\oplus$ is a generalization of addition, and $\otimes$ is a generalization of multiplication, with

identity elements **0** and **1** respectively. Table 1 shows several semirings with different usages: The real semiring is conventional math, the Boolean is symbolic logic. The tropical [13] and max-plus [1] semirings are closely related in that their domains can be though of as the negative logarithms and logarithms of the real numbers respectively. The probability semiring is the real semiring over the domain of valid probabilities, and the Viterbi semiring—of interest to this paper—is the same but with addition replaced with the maxoperator. In practice, we rarely deal with probabilities in the real domain due to underflow problems, but instead with (negative) log probabilities that rely on the logadd operator: $\text{logadd}(x, y) \equiv \log_b(b^x + b^y)$. The corresponding (negative) log Viterbi semiring replaces the logadd-operator with the (min) max operator. The choice between raw logarithms and negated logarithms is mostly a matter of taste, but there is a nice interpretation of negative log probabilities as costs that are higher for unlikely events and lower for likely events.

Intuitively, deductive systems and semirings can be understood as a generalization of Boolean logic with arbitrary **values** instead of *true*/*false*. We can use the generalized semiring operators to compute the value α, corresponding to inside/forward probabilities of a conclusion, as:

$$\alpha(b) = \bigoplus_{\frac{a_0,\cdots,a_{I-1}}{b}} \bigotimes_{i=0}^{I-1} \alpha(a_i) \tag{9}$$

One advantage with this generalization is that **reverse values**, βs, corresponding to backward/outside probabilities, can be derived from values:

$$\beta(a_i) \bigoplus_{\frac{a_0,\cdots,a_i,\cdots,a_{I-1}}{b}} \beta(b) \otimes \bigotimes_{j=0}^{I-1} \begin{cases} \alpha(a_j) & \text{if } i \neq j \\ \mathbf{1} & \text{otherwise} \end{cases} \tag{10}$$

where the reverse value of the goal item is assumed to be **1**.

2.4 Viterbi Training with Equiprobable Derivations

Viterbi training relies on finding the best derivation of a deductive system, and counting each rule used in that derivation once. Finding only the best derivation is more efficient because more efficient search algorithms can be used, and because, when operating in log domain (which is typically necessary to avoid underflow problems), the operations themselves are faster: the logadd procedure requires calls to log and exp, both relatively heavy functions, whereas max and min are built-in CPU instructions. One problem with Viterbi training that is often glossed over is that grammars frequently contain ambiguity, and there is a real risk of having multiple equiprobable best derivations. In practice, implementations typically solve this by choosing one to be the best on an *ad-hoc* basis, although it is understood that the choice should be random in order to avoid biasing the training. With enough observations any true ambiguity will be preserved by the random selection. This sounds good until you realize that counts

are integers, and that observing a phenomenon an uneven number of times will drive likelihood of the unfavored interpretation to zero in the next iteration.

Imagine for example the Swedish noun *stegen*, which has two interpretations: *steg+en* 'the steps' and *stege+n* 'the ladder'. Suppose that we have a CFG for noun phrases containing the following eight rules with their associated probabilities initialized to be an informative example:

$$\begin{aligned}
&\dots \\
&p\left(\mathrm{NP} \to \mathrm{NN}^{\mathrm{SG_DET}}\right) &&= 0.05 \\
&p\left(\mathrm{NP} \to \mathrm{NN}^{\mathrm{PL_DET}}\right) &&= 0.05 \\
&p\left(\mathrm{NP} \to \mathrm{Art}^{\mathrm{SG}}\mathrm{NN}^{\mathrm{SG_DET}}\right) &&= 0.15 \\
&p\left(\mathrm{NP} \to \mathrm{Art}^{\mathrm{PL}}\mathrm{NN}^{\mathrm{PL_DET}}\right) &&= 0.10 \\
&p\left(\mathrm{NN}^{\mathrm{SG_DET}} \to \textit{stegen}\right) &&= 0.03 \\
&p\left(\mathrm{NN}^{\mathrm{PL_DET}} \to \textit{stegen}\right) &&= 0.03 \\
&p\left(\mathrm{Art}^{\mathrm{SG}} \to \textit{den}\right) &&= 0.50 \\
&p\left(\mathrm{Art}^{\mathrm{PL}} \to \textit{de}\right) &&= 1.00 \\
&\dots
\end{aligned} \tag{11}$$

Suppose also that the grammar is ambiguous, so that $p\left(\mathrm{NP} \to \mathrm{NN}^{\mathrm{SG\_DET}}\right) = p\left(\mathrm{NP} \to \mathrm{NN}^{\mathrm{PL\_DET}}\right)$ and $p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \to \textit{stegen}\right) = p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \to \textit{stegen}\right)$. Our training data contains many example of noun phrases with articles, but only one example of a lone noun: *stegen*. Even if the ties are broken randomly, we will only ever get counts, and thus nonzero value for either $\mathrm{NP} \to \mathrm{NN}^{\mathrm{SG\_DET}}$ and $\mathrm{NN}^{\mathrm{SG\_DET}} \to \textit{stegen}$, or $\mathrm{NP} \to \mathrm{NN}^{\mathrm{PL\_DET}}$ and $\mathrm{NN}^{\mathrm{PL\_DET}} \to \textit{stegen}$, not both; the information that *stegen* is ambiguous will be lost. Now suppose our test data contains the noun phrases *de stegen* 'those steps' and *den stegen* 'that ladder'; we will only be able to assign a nonzero probability to one or the other, depending on whether we trained the grammar to treat *stegen* as a singular or plural noun. In this paper we will show that automatic differentiation under the Viterbi semiring preserves this type of ambiguity provided that an unbiased subderivative of max is used.

2.5 Automatic Differentiation

As pointed out in Li and Eisner [8] and further explored in Smith [14] it is possible to view the expectation step in expectation maximization as a case of automatic differentiation in the reverse mode [2], which makes it essentially identical to backpropagation [10], but over a lattice/forest rather than a neural network [4]. With this view, reverse values of rules are equivalent to partial derivatives of the goal value with respect to rule probabilities:

$$\beta\left(r\right) = \frac{\partial \alpha\left(S_{0,T}\right)}{\partial \alpha\left(r\right)} \tag{12}$$

We can use the chain rule to aggregate the derivatives of all consequences that any one condition can lead to:

$$\frac{\partial\alpha\left(S_{0,T}\right)}{\partial\alpha\left(a_i\right)} = \sum_{e=\frac{a_0,\cdots,a_i,\cdots,a_{I-1}}{b}} \frac{\partial\alpha\left(S_{0,T}\right)}{\partial\alpha\left(b\right)} \frac{\partial \sum_{e'=\frac{a_0,\cdots,a_{K-1}}{b}} \alpha\left(e'\right)}{\partial\alpha\left(e\right)} \frac{\partial \prod_{j=0}^{I-1} \alpha\left(a_j\right)}{\partial\alpha\left(a_i\right)} \tag{13}$$

which calculates the same quantity as traditional reverse values, which we get by substituting sum and product for their generalized place holders in Eq. 10:

$$\beta\left(a_i\right) = \sum_{\frac{a_0,\cdots,a_i,\cdots,a_{I-1}}{b}} \beta\left(b\right) \prod_{j=0}^{I-1} \begin{cases} \alpha\left(a_j\right) & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases} \tag{14}$$

The equality in Eq. 12 can be understood by inspecting the summation in Eq. 13 and noting that (a) the first factor is $\beta\left(b\right)$ provided that the equality holds (equal to the first factor in the summation in Eq. 14), (b) the second factor will always be 1 since it is the derivative of a sum with respect to one of its terms, and that (c) the third factor is the derivative of a product with respect to one of its factors, which is equivalent to the product of all the other factors (equal to the second factor in the summation in Eq. 14).

Relating this back of EM reestimation, we can reformulate Eq. 6 as:

$$q\left(r\right) = p\left(r\right) \sum_{w_{0..T} \in \mathcal{D}} \frac{1}{\alpha\left(S_{0,T}\right)} \frac{\partial\alpha\left(S_{0,T}\right)}{\partial\alpha\left(r\right)} \tag{15}$$

3 Reverse Values ≠ Derivatives

As we saw in Sect. 2.5, both reverse values and automatic differentiation can be used in EM-training, as they are equivalent under the real semiring; in this section we show that this equivalence fails to hold for the Viterbi semiring. First, we observe that values under the Viterbi semiring correspond to the most likely path/tree leading to a particular conclusion, so the value calculated for the goal item is the probability of the best path/tree. Next, we construct an expression for the derivative of a condition under the Viterbi semiring. Following Eq. 13 we get:

$$\frac{\partial\alpha\left(S_{0,T}\right)}{\partial\alpha\left(a_i\right)} = \sum_{e=\frac{a_0,\cdots,a_i,\cdots,a_{I-1}}{b}} \frac{\partial\alpha\left(S_{0,T}\right)}{\partial\alpha\left(b\right)} \frac{\partial \max_{e'=\frac{a_0,\cdots,a_{K-1}}{b}} \alpha\left(e'\right)}{\partial\alpha\left(e\right)} \frac{\partial \prod_{j=0}^{I-1} \alpha\left(a_j\right)}{\partial\alpha\left(a_i\right)} \tag{16}$$

Note that only the inner sum is replaced by a max, the outer sum is part of how derivatives are accumulated and has nothing to do with the semiring operators. Further note that we use max as an iterated binary operator like sum or product.

If we instead substitute the generalized operators in Eq. 10 with the Viterbi operators max and product we get:

$$\beta\left(a_i\right) = \max_{\frac{a_0,\cdots,a_i,\cdots,a_{I-1}}{b}} \beta\left(b\right) \prod_{j=0}^{I-1} \begin{cases} \alpha\left(a_j\right) & \text{if } i \neq j \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

Equartions 16 and 17 have very different forms, and there is no reason to believe that they would calculate the same quantity. The easiest way to see why they are different is to note that the second factor in the summation in Eq. 16 allows the entire term to become zero (whenever there is a competing way to arrive at b that has higher probability), whereas Eq. 17 is nonzero for all axioms that are used in the derivation.

4 Viterbi Training with Derivatives

Reverse values are unrelated to the rule counts needed for Viterbi training, but we show in this section that derivatives can be used to calculate counts for Viterbi training, and that the mechanism for doing so is the same as for how fractional counts but in a different semiring.

In the case when there is exactly one best derivation, its probability $\alpha(S_{0,T})$ is the product of the probability of all grammar rules used in that derivation. Designating the rules used in a derivation of sentence $w_{0..T}$ as $r_0, r_1, \cdots, r_k$we have:

$$c_{w_{0..T}}(r_i) = p(r_i) \frac{1}{\alpha(S_{0,T})} \frac{\partial \alpha(S_{0,T})}{\partial p(r_i)} = \frac{p(r_i)}{\prod_{j=0}^{k-1} p(r_j)} \frac{\partial \prod_{j=0}^{k-1} p(r_j)}{\partial p(r_i)} = \frac{p(r_i)}{p(r_i)} = 1 \tag{18}$$

which is what we would expect: every rule in the best derivation of a single sentence gets a count of one. More generally, the rule counts have the exact same form as the fractional counts in EM, but under the Viterbi semiring rather than the probability semiring:

$$c(r) = p(r) \sum_{w_{0..T} \in \mathcal{D}} \frac{1}{\alpha(S_{0,T})} \frac{\partial \alpha(S_{0,T})}{\partial \alpha(r)} \tag{19}$$

Calculating the derivatives of the max operator does, however, pose a problem for sentences where there are multiple equiprobable (sub-)derivations because the derivative of max is undefined for equal arguments. We can get around this by using the *sub*derivative, which allows us to distribute the results between the two maximizers. But with the subderivative we have to decide *how* to distribute the results between the two maximizers. We can characterize the distribution of the results with a parameter $0 \leq \lambda \leq 1$, and define our subderivative as:

$$\frac{\partial \max(x,y)}{\partial x} = \begin{cases} 1 \text{ if } & x > y \\ \lambda \text{ if } & x = y \\ 0 \text{ if } & x < y \end{cases} \text{ and } \frac{\partial \max(x,y)}{\partial y} = \begin{cases} 1 & \text{if } x > y \\ 1-\lambda \text{ if } & x = y \\ 0 & \text{if } x < y \end{cases} \tag{20}$$

5 Example

Only an unbiased subderivative for max will preserve genuine ambiguity found in the data, and to show the mechanism, we will return to our example CFG from 11. We have two derivations for the Swedish noun phrase *stegen*:

$$\frac{\frac{p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}\right)}{\mathrm{NN}_{0,1}^{\mathrm{SG\_DET}}}, p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}\right)}{\mathrm{NP}_{0,1}} \tag{21}$$

and

$$\frac{\frac{p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right)}{\mathrm{NN}_{0,1}^{\mathrm{PL\_DET}}}, p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{PL\_DET}}\right)}{\mathrm{NP}_{0,1}} \tag{22}$$

The value of the noun phrase $\mathrm{NP}_{0,1}$ is:

$$\alpha\left(\mathrm{NP}_{0,1}\right) = \max\left(\begin{array}{c} p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}\right) p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}\right), \\ p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{PL\_DET}}\right) p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right) \end{array}\right) \tag{23}$$

where both arguments to max are equal, which is the property that we want to preserve, since the data lacks any evidence for preferring one over the other. We can work out the counts of a rule $r_1 = \mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}$ with its companion rule $r_2 = \mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}$ as follows:

$$\begin{aligned} c\left(r_1\right) &= p\left(r_1\right) \frac{1}{\alpha\left(\mathrm{NP}_{0,1}\right)} \frac{\partial \alpha\left(\mathrm{NP}_{0,1}\right)}{\partial p\left(r_1\right)} && (24) \\ &= p\left(r_1\right) \frac{1}{\alpha\left(\mathrm{NP}_{0,1}\right)} \frac{\partial \alpha\left(\mathrm{NP}_{0,1}\right)}{\partial p\left(r_1\right) p\left(r_2\right)} \frac{\partial p\left(r_1\right) p\left(r_2\right)}{\partial p\left(r_1\right)} && (25) \\ &= p\left(r_1\right) \frac{1}{p\left(r_1\right) p\left(r_2\right)} \lambda p\left(r_2\right) && (26) \\ &= \lambda && (27) \end{aligned}$$

conversely for the other rules we have:

$$\begin{aligned} c\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}\right) &= \lambda, && (28) \\ c\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{PL\_DET}}\right) &= 1 - \lambda, && (29) \\ c\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right) &= 1 - \lambda && (30) \end{aligned}$$

as we can see, setting $\lambda = 0$ or $\lambda = 1$ recreates the behavior of choosing one or the other, whereas setting $\lambda = 0.5$ preserves the desired ambiguity. Since setting λ to anything but 0.5 will cause the winner to be sole maximizer in the next iteration of training, any value other than 0.5 has the same effect as setting it to 0 or 1. We call the subderivative that preserve the ambiguity **unbiased**.

After one iteration of training, the rules of interest would have their probabilities altered depending on λ such that:

$$\begin{aligned} \lambda = 1 \qquad & \qquad \lambda = 0.5 \\ p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}\right) = 0.10 \qquad & \qquad p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{SG\_DET}}\right) = 0.05 \\ p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{PL\_DET}}\right) = 0.00 \qquad & \qquad p\left(\mathrm{NP} \rightarrow \mathrm{NN}^{\mathrm{PL\_DET}}\right) = 0.05 \\ p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}\right) = 0.06 \qquad & \qquad p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}\right) = 0.03 \\ p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right) = 0.00 \qquad & \qquad p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right) = 0.03 \end{aligned}$$

Looking at the derivations of our test sentences, the grammar trained with the unbiased subderivative give non-zero probabilities to both *de stegen* 'those steps':

$$\frac{\frac{p\left(\mathrm{Art}^{\mathrm{PL}} \rightarrow \textit{de}\right)}{\mathrm{Art}_{0,1}^{\mathrm{PL}}}, \frac{p\left(\mathrm{NN}^{\mathrm{PL\_DET}} \rightarrow \textit{stegen}\right)}{\mathrm{NN}_{1,2}^{\mathrm{PL\_DET}}}, p\left(\mathrm{NP} \rightarrow \mathrm{Art}^{\mathrm{PL}}\mathrm{NN}^{\mathrm{PL\_DET}}\right)}{\mathrm{NP}_{0,2}} \tag{31}$$

and *den stegen* 'that ladder':

$$\frac{\frac{p\left(\mathrm{Art}^{\mathrm{SG}} \rightarrow \textit{den}\right)}{\mathrm{Art}_{0,1}^{\mathrm{SG}}}, \frac{p\left(\mathrm{NN}^{\mathrm{SG\_DET}} \rightarrow \textit{stegen}\right)}{\mathrm{NN}_{1,2}^{\mathrm{SG\_DET}}}, p\left(\mathrm{NP} \rightarrow \mathrm{Art}^{\mathrm{SG}}\mathrm{NN}^{\mathrm{SG\_DET}}\right)}{\mathrm{NP}_{0,2}} \tag{32}$$

whereas the biased subderivative assigns zero probability to derivation Eq. 31.

The unbiased subderivative of max has a problem when being generalized to iterated binary operations. Consider $\max(a, b, c)$ where $a = b = c$. If we binarize it as $\max(a, \max(b, c))$,

$$\frac{\partial \max(a, \max(b, c))}{\partial a} = \lambda \neq \frac{\partial \max(a, \max(b, c))}{\partial b} = \lambda(1 - \lambda) \tag{33}$$

Luckily, we can generalize the unbiased subderivative to distribute the mass uniformly to one or more arguments:

$$\frac{\partial \max_{k=0}^{K-1} x_k}{\partial x_i} = \begin{cases} \dfrac{1}{\sum_{j=0}^{K-1} \begin{cases} 1 \text{ if } x_j = \max_{k=0}^{K-1} x_k \\ 0 \text{ otherwise} \end{cases}} & \text{if } x_i = \max_{k=0}^{K-1} x_k \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

The time complexity for this derivative of max is still linear in K.

6 With Respect to the Entire Data Set

It turns out that the derivatives have another advantage, as we can generalize the calculation away from individual sentences and apply them to the entire data

set; specifically, the (fractional) counts are obtained when taking the partial derivative of rule probabilities with respect to the logarithm of the probability of the data:

$$p(r)\frac{\partial \log \prod_{w_{0..T}\in\mathcal{D}} \alpha(S_{0,T})}{\partial p(r)} = p(r)\frac{\partial \sum_{w_{0..T}\in\mathcal{D}} \log\alpha(S_{0,T})}{\partial p(r)} \tag{35}$$

$$= p(r)\sum_{w_{0..T}\in\mathcal{D}} \frac{\partial \log\alpha(S_{0,T})}{\partial p(r)} \tag{36}$$

$$= p(r)\sum_{w_{0..T}\in\mathcal{D}} \frac{\partial \log\alpha(S_{0,T})}{\partial \alpha(S_{0,T})}\frac{\partial \alpha(S_{0,T})}{\partial p(r)} \tag{37}$$

$$= p(r)\sum_{w_{0..T}\in\mathcal{D}} \frac{1}{\alpha(S_{0,T})}\frac{\partial \alpha(S_{0,T})}{\partial p(r)} \tag{38}$$

$$= c(r) \tag{39}$$

We get the counts needed for Viterbi training when differentiating under the Viterbi semiring, and the fractional counts or sufficient statistics needed for expectation maximization when differentiating under the probability semiring. This gives an interesting unified view of training as applying maximum likelihood estimation to "expected rule probability gradient".

7 Conclusions

We have showed that the equality between a reverse pass and automatic differentiation, which exists for the probability semiring, fails to hold for the Viterbi semiring, and that automatic differentiation, in contrast to reverse values, gives the counts needed for Viterbi training. The differentiation approach is the same for both expectation maximization and Viterbi training except for the semiring used, opening up great opportunities for code reuse in implementations. We further highlighted a problem with the intuitive way of doing Viterbi training, in that ambiguous models, whose best paths/trees should be counted proportionally to not loose important information, are hard to count correctly and efficiently; again, just using automatic differentiation solves the problem, provided that an unbiased subderivative of max is used.

Acknowledgements. This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under LORELEI contract HR0011-15-C-0114, BOLT contracts HR0011-12-C-0014 and HR0011-12-C-0016, and GALE contracts HR0011-06-C-0022 and HR0011-06-C-0023; by the European Union under the Horizon 2020 grant agreement 645452 (QT21) and FP7 grant agreement 287658; and by the Hong Kong Research Grants Council (RGC) research grants GRF16210714, GRF16214315, GRF620811 and GRF621008. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the EU, or RGC. The authors would also like to thank the anonymous reviewers for valuable feedback.

References

1. Baccelli, F., Cohen, G., Older, G.J., Quadrat, J.P.: Synchronization and Linearity: An Algebra For Discrete Event Systems. Wiley Series in Probability and Mathematical Statistics. Wiley, Chichester (1992)
2. Corliss, G., Faure, C., Griewank, A., Hascoët, L., Naumann, U. (eds.): Automatic Differntiation of Algorithms: From Simulation to Optimization. Springer, New York (2002). https://doi.org/10.1007/978-1-4613-0075-5
3. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol.) **39**(1), 1–38 (1977)
4. Eisner, J.: Inside-outside and forward-backward algorithms are just backprop. In: Proceedings of the EMNLP Workshop on Structured Prediction for NLP, Austin, Texas, November 2016
5. Eisner, J., Goldlust, E., Smith, N.A.: Compiling comp Ling: weighted dynamic programming and the Dyna language. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP), Vancouver, Canada, pp. 281–290, October 2005
6. Goodman, J.: Semiring parsing. Comput. Linguist. **25**(4), 573–605 (1999)
7. Juang, B.H., Rabiner, L.R.: The segmental K-means algorithm for estimating parameters of hidden Markov models. IEEE Trans. Acoust. Speech Signal Process. **38**, 1639–1641 (1990)
8. Li, Z., Eisner, J.: First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In: 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009), Singapore, pp. 40–51, August 2009
9. Pereira, F.C.N., Warren, D.H.D.: Parsing as deduction. In: Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL 1983), Cambridge, Massachusetts, pp. 137–144, June 1983
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
11. Saers, M., Wu, D.: Reestimation of reified rules in semiring parsing and biparsing. In: Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-5), Portland, Oregon, pp. 70–78, June 2011
12. Shieber, S.M., Schabes, Y., Pereira, F.C.: Principles and implementation of deductive parsing. J. Logic Program. **24**(1–2), 3–36 (1995)
13. Simon, I.: Recognizable sets with multiplicities in the tropical semiring. In: Chytil, M.P., Koubek, V., Janiga, L. (eds.) MFCS 1988. LNCS, vol. 324, pp. 107–120. Springer, Heidelberg (1988). https://doi.org/10.1007/BFb0017135
14. Smith, N.A.: Linguistic structure prediction. Synth. Lect. Hum. Lang. Technol. **4**(2), 1–274 (2011)