Inference algorithms for pattern-based CRFs on sequence data

Rustem Takhanov TAKHANOV@MAIL.RU
Institute of Science and Technology (IST), Austria

Vladimir Kolmogorov VNK@IST.AC.AT
Institute of Science and Technology (IST), Austria

Abstract

We consider *Conditional Random Fields (CRFs) with pattern-based potentials* defined on a chain. In this model the energy of a string (labeling) $x_1 \ldots x_n$ is the sum of terms over intervals $[i, j]$ where each term is non-zero only if the substring $x_i \ldots x_j$ equals a prespecified pattern α. Such CRFs can be naturally applied to many sequence tagging problems.

We present efficient algorithms for the three standard inference tasks in a CRF, namely computing (i) the partition function, (ii) marginals, and (iii) computing the MAP. Their complexities are respectively $O(nL)$, $O(nL\ell_{\max})$ and $O(nL \min\{|D|, \log(\ell_{\max}+1)\})$ where L is the combined length of input patterns, $\ell_{\max}$ is the maximum length of a pattern, and D is the input alphabet. This improves on the previous algorithms of (Ye et al., 2009) whose complexities are respectively $O(nL|D|)$, $O\left(n|\Gamma|L^2\ell_{\max}^2\right)$ and $O(nL|D|)$, where $|\Gamma|$ is the number of input patterns. In addition, we give an efficient algorithm for sampling, and revisit the case of MAP with non-positive weights. Finally, we apply pattern-based CRFs to the problem of the protein dihedral angles prediction.

1. Introduction

This paper addresses the *sequence labeling* (or the *sequence tagging*) problem: given an observation z (which is usually a sequence of n values), infer labeling $x = x_1 \ldots x_n$ where each variable x_i takes values in some finite domain D. Such problem appears in many domains such as text and speech analysis, signal analysis, and bioinformatics.

One of the most successful approaches for tackling the problem is the Hidden Markov Model (HMM). The kth order HMM is given by the probability distribution $p(x|z) = \frac{1}{Z}\exp\{-E(x|z)\}$ with the energy function

$$E(x|z) = \sum_{i\in[1,n]} \psi_i(x_i, z_i) + \sum_{(i,j)\in\mathcal{E}_k} \psi_{ij}(x_{i:j}) \qquad (1)$$

where $\mathcal{E}_k = \{(i, i+k)\,|\,i \in [1, n-k]\}$ and $x_{i:j} = x_i \ldots x_j$ is the substring of x from i to j. A popular generalization is the Conditional Random Field model (Lafferty et al., 2001) that allows all terms to depend on the full observation z:

$$E(x|z) = \sum_{i\in[1,n]} \psi_i(x_i, z) + \sum_{(i,j)\in\mathcal{E}_k} \psi_{ij}(x_{i:j}, z) \qquad (2)$$

We study a particular variant of this model called a *pattern-based CRF*. It is defined via

$$E(x|z) = \sum_{\alpha\in\Gamma} \sum_{\substack{[i,j]\subseteq[1,n] \\ j-i+1=|\alpha|}} \psi_{ij}^{\alpha}(z) \cdot [x_{i:j} = \alpha] \qquad (3)$$

where Γ is a fixed set of non-empty words, $|\alpha|$ is the length of word α and $[\cdot]$ is the *Iverson bracket*. If we take $\Gamma = D \cup D^k$ then (3) becomes equivalent to (2); thus we do not loose generality (but gain more flexibility).

Intuitively, pattern-based CRFs allow to model long-range interactions for selected subsequences of labels. This could be useful for a variety of applications: in part-of-speech tagging patterns could correspond to certain syntactic constructions or stable idioms; in protein secondary structure prediction - to sequences of dihedral angles corresponding to stable configuration such as α-helixes; in gene prediction - to sequences of nucleatydes with supposed functional roles such as “exon” or “intron”, specific codons, etc.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

Inference This paper focuses on inference algorithms for pattern-based CRFs. The three standard inference tasks are (i) computing the partition function $Z = \sum_x \exp\{-E(x|z)\}$; (ii) computing marginal probabilities $p(x_{i:j} = \alpha|z)$ for all triplets (i,j,α) present in (3); (iii) computing MAP, i.e. minimizing energy (3). The complexity of solving these tasks is discussed below. We denote $L = \sum_{\alpha\in\Gamma} |\alpha|$ to be total length of patterns and $\ell_{\max} = \max_{\alpha\in\Gamma} |\alpha|$ to be the maximum length of a pattern.

A naive approach is to use standard message passing techniques for an HMM of order $k = \ell_{\max} - 1$. However, they would take $O(n|D|^{k+1})$ time which would become impractical for large k. More efficient algorithms with complexities $O(nL|D|)$, $O\left(n|\Gamma|L^2\ell_{\max}^2\right)$ and $O(nL|D|)$ respectively were given by (Ye et al., 2009).[1] Our first contribution is to improve this to $O(nL)$, $O(nL\ell_{\max})$ and $O(nL \cdot \min\{|D|, \log(\ell_{\max}+1)\})$ respectively (more accurate estimates are given in the next section).

We also give an algorithm for sampling from the distribution $p(x|z)$. Its complexity is either (i) $O(nL)$ per sample, or (ii) $O(n)$ per sample with an $O(nL|D|)$ preprocessing (assuming that we have an oracle that produces independent samples from the uniform distribution on $[0,1]$ in $O(1)$ time).

Finally, we consider the case when all costs $\psi_{ij}^{\alpha}(z)$ are non-positive. (Komodakis & Paragios, 2009) gave an $O(nL)$ technique for minimizing energy (3) in this case. We present a modification that has the same worst-case complexity but can beat the algorithm of (Komodakis & Paragios, 2009) in the best case.

Related work The works of (Ye et al., 2009) and (Komodakis & Paragios, 2009) are probably the most related to our paper. The former applied pattern-based CRFs to the handwritten character recognition problem and to the problem of identification of named entities from texts. The latter considered a pattern-based CRF on a grid for a computer vision application; the MAP inference problem in (Komodakis & Paragios, 2009) was converted to sequence labeling problems by decomposing the grid into thin "stripes".

(Qian et al., 2009) considered a more general formulation in which a single pattern is characterized by a set of strings rather than a single string α. They proposed an exact inference algorithm and applied it to the OCR task and to the Chinese Organization Name Recognition task. However, their algorithm could take time exponential in the total lengths of input patterns; no subclasses of inputs were identified which could be solved in polynomial time.

A different generalization (for non-sequence data) was proposed in (Rother et al., 2009). Their inference procedure reduces the problem to the MAP estimation in a pairwise CRF with cycles, which is then solved with approximate techniques such as BP, TRW or QPBO. This model was applied to the texture restoration problem.

(Nguyen et al., 2011) extended algorithms in (Ye et al., 2009) to the *Semi-Markov model* (Sarawagi & Cohen, 2004). We conjecture that our algorithms can be extended to this case as well, and can yield a better complexity compared to (Nguyen et al., 2011).

2. Notation and preliminaries

First, we introduce a few definitions.

- A *pattern* is a pair $\alpha = ([i,j], x)$ where $[i,j]$ is an interval in $[1,n]$ and $x = x_i \dots x_j$ is a sequence over alphabet D indexed by integers in $[i,j]$ $(j \geq i-1)$. The *length* of α is denoted as $|\alpha| = |x| = j - i + 1$.
- Symbols "$*$" denotes an arbitrary word or pattern (possibly the empty word ε or the empty pattern $\varepsilon_s \triangleq ([s+1, s], \varepsilon)$ at position s). The exact meaning will always be clear from the context. Similary, "$+$" denotes an arbitrary non-empty word or pattern.
- The concatenation of patterns $\alpha = ([i,j], x)$ and $\beta = ([j+1,k], y)$ is the pattern $\alpha\beta \triangleq ([i,k], xy)$. Whenever we write $\alpha\beta$ we assume that it is defined, i.e. $\alpha = ([\cdot, j], \cdot)$ and $\beta = ([j+1, \cdot], \cdot)$ for some j.
- For a pattern $\alpha = ([i,j], x)$ and interval $[k,\ell] \subseteq [i,j]$, the *subpattern of* α *at position* $[k,\ell]$ is the pattern $\alpha_{k:\ell} \triangleq ([k,\ell], x_{k:\ell})$ where $x_{k:\ell} = x_k \dots x_\ell$.
  If $k = i$ then $\alpha_{k:\ell}$ is called a *prefix* of α. If $\ell = j$ then $\alpha_{k:\ell}$ is a *suffix* of α.
- If β is a subpattern of α, i.e. $\beta = \alpha_{k:\ell}$ for some $[k,\ell]$, then we say that β is *contained* in α. This is equivalent to the condition $\alpha = *\beta*$.
- $D^{i:j} = \{([i,j], x) \mid x \in D^{[i,j]}\}$ is the set of patterns with interval $[i,j]$. We typically use letter x for patterns in $D^{1:s}$ and letters $\alpha, \beta, \dots$ for other patterns. Patterns $x \in D^{1:s}$ will be called *partial labelings*.
- For a set of patterns Π and index $s \in [0,n]$ we denote Π_s to be the set of patterns in Π that end at position s: $\Pi_s = \{([i,s], \alpha) \in \Pi\}$.
- For a pattern α let α^- be the prefix of α of length $|\alpha| - 1$; if α is empty then α^- is undefined.

We will consider the following general problem. Let Π° be the set of patterns of words in Γ placed at all

[1]Some of the bounds stated in (Ye et al., 2009) are actually weaker. However, it is not difficult to show that their algorithms can be implemented in times stated above, using our Lemma 1.

possible positions: $\Pi^\circ = \{([i,j],\alpha) \mid \alpha \in \Gamma)\}$. Let $(R, \oplus, \otimes)$ be a commutative semiring with elements $\mathbb{0}, \mathbb{1} \in R$ which are identities for $\oplus$ and $\otimes$ respectively. Define the cost of pattern $x \in D^{i:j}$ via

$$f(x) = \bigotimes_{\alpha \in \Pi^\circ, x = *\alpha*} c_\alpha \tag{4}$$

where $c_\alpha \in R$ are fixed constants. (Throughout the paper we adopt the convention that operations $\oplus$ and $\otimes$ over the empty set of arguments give respectively $\mathbb{0}$ and $\mathbb{1}$, and so e.g. $f(\varepsilon_s) = \mathbb{1}$.) Our goal is to compute

$$Z = \bigoplus_{x \in D^{1:n}} f(x) \tag{5}$$

Example 1 *If $(R, \oplus, \otimes) = (\mathbb{R}, +, \times)$ then problem (5) is equivalent to computing the partition function for the energy (3), if we set $c_{([i,j],\alpha)} = \exp\{-\psi^\alpha_{ij}(z)\}$.*

Example 2 *If $(R, \oplus, \otimes) = (\overline{\mathbb{R}}, \min, +)$ where $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{+\infty\}$ then we get the problem of minimizing energy (3), if $c_{([i,j],\alpha)} = \psi^\alpha_{ij}(z)$.*

The complexity of our algorithms will be stated in terms of the following quantities:

- $P = |\{\alpha \mid \exists \alpha* \in \Gamma, \alpha \neq \varepsilon\}|$ is the number of distinct non-empty prefixes of words in Γ. Note that $P \leq L$.
- $P' = |\{\alpha | \exists \alpha+ \in \Gamma\}|$ is the number of distinct proper prefixes of words in Γ. There holds $\frac{P}{P'} \in [1, |D|]$. If $\Gamma = D^1 \cup D^2 \cup \ldots \cup D^k$ then $\frac{P}{P'} = |D|$. If Γ is a sparse random subset of the set above then $\frac{P}{P'} \approx 1$.
- $I(\Gamma) = \{\alpha \mid \exists \alpha*, *\alpha \in \Gamma, \alpha \neq \varepsilon\}$ is the set of non-empty words which are both prefixes and suffixes of some words in Γ. Note that $\Gamma \subseteq I(\Gamma)$ and $|I(\Gamma)| \leq P$.

We will present 6 algorithms (omitting proofs due to space limitations; **all proofs are given in (Takhanov & Kolmogorov, 2012)**):

Sec. 3: $\Theta(nP)$ algorithm for the case when $(R, \oplus, \otimes)$ is a ring, i.e. it has operation $\ominus$ that satisfies $(a \ominus b) \oplus b = a$ for all $a, b \in R$. This holds for the semiring in Example 1 (but not for Example 2).
Sec. 4: $\Theta(nP)$ algorithm for sampling. Alternatively, it can be implemented to produce independent samples in $\Theta(n)$ time per sample with a $\Theta(nP|D|)$ preprocessing.
Sec. 5: $O(n \sum_{\alpha \in I(\Gamma)} |\alpha|)$ algorithm for computing marginals for all patterns $\alpha \in \Pi^\circ$.
Sec. 6: $\Theta(nP'|D|)$ algorithm for a general commutative semiring, which is equivalent to the algorithm in (Ye et al., 2009). It will be used as a motivation for the next algorithm.
Sec. 7: $O(nP \log P)$ algorithm for a general commutative semiring; for the semiring in Example 2 the complexity can be reduced to $O(nP \log(\ell_{\max} + 1))$.

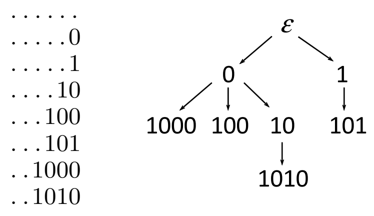

Figure 1. Graph $G[\Pi_s]$ for the set of 8 patterns shown on the left (for brevity, their intervals are not shown; they all end at the same position s.) This set of patterns would arise if $\Gamma = \{0, 1, 1000, 1010\}$ and Π was defined as the set of all prefixes of patterns in Π°.

Sec. 8 in (Takhanov & Kolmogorov, 2012): $O(nP)$ algorithm for the case $(R, \oplus, \otimes) = (\overline{\mathbb{R}}, \min, +)$, $c_\alpha \leq 0$ for all $\alpha \in \Pi^\circ$.

All algorithms will have the following structure. Given the set of input patterns Π°, we first construct another set of patterns Π; it will typically be either the set of prefixes or the set of proper prefixes of patterns in Π°. This can be done in a preprocessing step since sets Π_s will be isomorphmic (up to a shift) for indexes s that are sufficiently far from the boundary. (Recall that Π_s is the set of patterns in Π that end at position s.) Then we recursively compute *messages* $M_s(\alpha)$ for $\alpha \in \Pi_s$ which have the following interpretation: $M_s(\alpha)$ is the sum ("$\oplus$") of costs $f(x)$ over a certain set of partial labelings of the form $x = *\alpha \in D^{1:s}$. In some of the algorithms we also compute messages $W_s(\alpha)$ which is the sum of $f(x)$ over **all** partial labelings of the form $x = *\alpha \in D^{1:s}$.

Graph $G[\Pi_s]$ The following construction will be used throughout the paper. Given a set of patterns Π and index s, we define $G[\Pi_s] = (\Pi_s, E[\Pi_s])$ to be a directed graph with the following set of edges: (α, β) belongs to $E[\Pi_s]$ for $\alpha, \beta \in \Pi_s$ if α is a proper suffix of β ($\beta = +\alpha$) and Π_s does not have an "intermediate" suffix γ of β with $|\beta| > |\gamma| > |\alpha|$. It can be checked that graph $G[\Pi_s]$ is a directed forest. If $\varepsilon_s \in \Pi_s$ then $G[\Pi_s]$ is connected and therefore is a tree. In this case we treat ε_s as the root. An example is shown in Fig. 1.

Computing partial costs Recall that $f(\alpha)$ for a pattern α is the cost of all patterns inside α (eq. (4)). We also define $\phi(\alpha)$ to be the cost of only those patterns that are suffixes of α:

$$\phi(\alpha) = \bigotimes_{\beta \in \Pi^\circ, \alpha = *\beta} c_\beta \tag{6}$$

Quantities $\phi(\alpha)$ and $f(\alpha)$ will be heavily used by the algorithms below; let us show how to compute them efficiently.

Lemma 1. *Let Π be a set of patterns with $\varepsilon_s \in \Pi$ for* *all $s \in [0,n]$. Values $\phi(\alpha)$ for all $\alpha \in \Pi$ can be computed using $O(|\Pi|)$ multiplications ("$\otimes$"). The same holds for values $f(\alpha)$ assuming that Π is prefix-closed, i.e. $\alpha^- \in \Pi$ for all non-empty patterns $\alpha \in \Pi$.*

Sets of partial labelings Let Π_s be a set of patterns that end at position s. Assume that $\varepsilon_s \in \Pi_s$. For a pattern $\alpha \in \Pi_s$ we define

$$\mathcal{X}_s(\alpha) = \{x \in D^{1:s} \mid x = *\alpha\} \quad (7)$$

$$\mathcal{X}_s(\alpha;\Pi_s) = \mathcal{X}_s(\alpha) - \bigcup_{(\alpha,\beta)\in E[\Pi_s]} \mathcal{X}_s(\beta) \quad (8)$$

It can be seen that sets $\mathcal{X}_s(\alpha;\Pi_s)$ are disjoint, and their union over $\alpha \in \Pi_s$ is $D^{1:s}$. Furthermore, there holds

$$\mathcal{X}_s(\alpha;\Pi_s) = \{x \in \mathcal{X}_s(\alpha) \mid x \neq *\beta \ \ \forall \beta = +\alpha \in \Pi_s\} \quad (9)$$

We will use eq. (9) as the definition of $\mathcal{X}_s(\alpha;\Pi_s)$ in the case when $\alpha \notin \Pi_s$.

3. Computing partition function

In this section we give an algorithm for computing quantity (5) assuming that $(R,\oplus,\otimes)$ is a ring. This can be used, in particular, for computing the partition function. We will assume that $D \subseteq \Gamma$; we can always add D to Γ if needed[2].

First, we select set Π as the set of prefixes of patterns in Π°:

$$\Pi = \{\alpha \mid \exists \alpha * \in \Pi^\circ\} \quad (10)$$

We will compute the following quantities for each $s \in [0,n]$, $\alpha \in \Pi_s$:

$$M_s(\alpha) = \bigoplus_{x\in\mathcal{X}_s(\alpha;\Pi_s)} f(x)\ , \quad W_s(\alpha) = \bigoplus_{x\in\mathcal{X}_s(\alpha)} f(x) \quad (11)$$

It is easy to see that for $\alpha \in \Pi_s$ the following equalities relate M_s and W_s:

$$M_s(\alpha) = W_s(\alpha) \ominus \bigoplus_{(\alpha,\beta)\in E[\Pi_s]} W_s(\beta) \quad (12a)$$

$$W_s(\alpha) = M_s(\alpha) \oplus \bigoplus_{(\alpha,\beta)\in E[\Pi_s]} W_s(\beta) \quad (12b)$$

[2]Note that we still claim complexity $O(nP)$ where P is the number of distinct non-empty prefixes of words in the *original* set Γ. Indeed, we can assume w.l.o.g. that each letter in D occurs in at least one word $\alpha \in \Gamma$. (If not, then we can "merge" non-occuring letters to a single letter and add this letter to Γ; clearly, any instance over the original pair (D,Γ) can be equivalenly formulated as an instance over the new pair. The transformation increases P only by 1). The assumption implies that $|D| \leq P$. Adding D to Γ increases P by at most P, and thus does not affect bound $O(nP)$.

These relations motivate the following algorithm. Since $|\Pi_s| = P+1$ for indexes s that are sufficiently far from the boundary, its complexity is $\Theta(nP)$ assuming that values $\phi(\alpha)$ in eq. (13a) are computed using Lemma 1.

Algorithm 1 Computing $Z = \bigoplus_{x\in D^{1:n}} f(x)$ for a ring

```
1: initialize messages: set $W_0(\varepsilon_0) := \mathbb{O}$
2: for each $s = 1,\ldots,n$ traverse nodes $\alpha \in \Pi_s$ of tree
   $G[\Pi_s]$ starting from the leaves and set

   $M_s(\alpha) := \phi(\alpha) \otimes \left[ W_{s-1}(\alpha^-) \ominus \bigoplus_{(\alpha,\beta)\in E[\Pi_s]} W_{s-1}(\beta^-) \right]$   (13a)
   $W_s(\alpha) := M_s(\alpha) \oplus \bigoplus_{(\alpha,\beta)\in E[\Pi_s]} W_s(\beta)$   (13b)

   Exception: if $\alpha = \varepsilon_s$ then set $M_s(\alpha) := \mathbb{O}$ instead
   of (13a)
3: return $Z := W_n(\varepsilon_n)$
```

Theorem 2. *Algorithm 1 is correct, i.e. it returns the correct value of $Z = \bigoplus_x F(x)$.*

4. Sampling

In this section consider the semiring $(R,\oplus,\otimes) = (\mathbb{R},+,\times)$ from Example 1. We assume that all costs c_α are strictly positive. We present an algorithm for sampling labelings $x \in D^{1:n}$ according to the probability distribution $p(x) = f(x)/Z$.

As in the previous section, we assume that $D \subseteq \Gamma$, and define Π to be the set of prefixes of patterns in Π° (eq. (10)). For a node $\alpha \in \Pi_s$ let $T_s(\alpha)$ be the set of nodes in the subtree of $G[\Pi_s]$ rooted at α, with $\alpha \in T_s(\alpha) \subseteq \Pi_s$. For a pattern $\alpha \in \Pi_{s+1} - \{\varepsilon_{s+1}\}$ we define set

$$\Delta_s(\alpha) = T_s(\alpha^-) - \bigcup_{(\alpha,\beta)\in G[\Pi_{s+1}]} T_s(\beta^-) \quad (14)$$

We can now present the algorithm.

Algorithm 2 Sample $x \sim p(x) = f(x)/Z$

```
1: run Algorithm 1 to compute messages $M_s(\alpha)$ for
   all patterns $\alpha = ([\cdot,s],\cdot) \in \Pi$
2: sample $\alpha_n \in \Pi_n$ with probability $p(\alpha_n) \propto M_n(\alpha_n)$
3: for $s = n-1,\ldots,1$ sample $\alpha_s \in \Delta_s(\alpha_{s+1})$ with
   probability $p(\alpha_s) \propto M_s(\alpha_s)$
4: return labeling $x$ with $x_{s:s} = (\alpha_s)_{s:s}$ for $s \in [1,n]$
```

We say that step s of the algorithm is *valid* if either (i) $s = n$, or (ii) $s \in [1,n-1]$, step $s+1$ is valid,

$\alpha_{s+1} \neq \varepsilon_{s+1}$ and $M_s(\alpha) > 0$ for some $\alpha \in \Delta_s(\alpha_{s+1})$. (This is a recursive definition.) Clearly, if step s is valid then line 3 of the algorithm is well-defined.

Theorem 3. *(a) With probability 1 all steps of the algorithm are valid. (b) The returned labeling $x \in D^{1:n}$ is distributed according to $p(x) = f(x)/Z$.*

Complexity Assume that we have an oracle that produces independent samples from the uniform distribution on $[0, 1]$ in $O(1)$ time.

The main subroutine performed by the algorithm is sampling from a given discrete distribution. Clearly, this can be done in $\Theta(N)$ time where N is the number of allowed values of the random variable. With a $\Theta(N)$ preprocessing, a sample can also be produced in $O(1)$ time by the so-called "alias method" (Vose, 1991).

This leads to two possible complexities: (i) $\Theta(nP)$ (without preprocessing); (ii) $\Theta(n)$ per sample (with preprocessing). Let us discuss the complexity of this preprocessing. Running Algorithm 1 takes $\Theta(nP)$ time. After that, for each $\alpha \in \Pi_{s+1}$ we need to run the linear time procedure of (Vose, 1991) for distributions $p(\beta) \propto M_s(\beta), \beta \in \Delta_s(\alpha_{s+1})$. The following theorem implies that this takes $\Theta(nP|D|)$ time.

Theorem 4. *There holds $\sum_{\alpha\in\Pi_s-\{\varepsilon_s\}} |\Delta_{s-1}(\alpha)| = |\Pi_{s-1}| \cdot |D|$.*

To summarize, we showed that with a $\Theta(nP|D|)$ preprocessing we can compute independent samples from $p(x)$ in $\Theta(n)$ time per sample.

5. Computing marginals

In this section we again consider the semiring $(R, \oplus, \otimes) = (\mathbb{R}, +, \times)$ from Example 1 where all costs c_α are strictly positive, and consider a probablity distribution $p(x) = f(x)/Z$ over labelings $x \in D^{1:n}$.

For a pattern α we define

$$\Omega(\alpha) = \{x \in D^{1:n} \mid x = *\alpha*\} \qquad (15)$$

$$Z(\alpha) = \sum_{x\in\Omega(\alpha)} f(x) \qquad (16)$$

We also define the set of patterns

$$\Pi = \{\alpha \mid \exists \alpha*, *\alpha \in \Pi^\circ, \alpha \text{ is non-empty}\} \qquad (17)$$

Note that $\Pi^\circ \subseteq \Pi$ and $|\Pi_s| = |I(\Gamma)|$ for indexes s that are sufficiently far from the boundary. We will present an algorithm for computing $Z(\alpha)$ for all patterns $\alpha \in \Pi$ in time $O(n \sum_{\alpha\in I(\Gamma)} |\alpha|)$. Marginal probabilities of a pattern-based CRF can then be computed as $p(x_{i:j} = \alpha) = Z(\alpha)/Z$ for a pattern $\alpha = ([i, j], \cdot)$.

In the previous section we used graph $G[\Pi_s]$ for a set of patterns Π_s; here we will need an analogous but a slightly different construction for patterns in Π. For patterns α, β we write $\alpha \sqsubseteq \beta$ if $\beta = *\alpha*$. If we have $\beta = +\alpha+$ then we write $\alpha \sqsubset \beta$.

Now consider $\alpha \in \Pi$. We define $\Phi(\alpha)$ to be the set of patterns $\beta \in \Pi$ such that $\alpha \sqsubset \beta$ and there is no other pattern $\gamma \in \Pi$ with $\alpha \sqsubset \gamma \sqsubseteq \beta$.

Our algorithm is given below. In the first step it runs Algorithm 1 from left to right and from right to left; as a result, we get forward messages $\overrightarrow{W}_j(\alpha)$ and backward messages $\overleftarrow{W}_i(\alpha)$ for patterns $\alpha = ([i, j], \cdot)$ such that

$$\overrightarrow{W}_j(\alpha) = \sum_{\substack{x=*\alpha \\ x\in D^{1:j}}} f(x) \qquad \overleftarrow{W}_i(\alpha) = \sum_{\substack{y=\alpha* \\ y\in D^{i:n}}} f(y) \qquad (18)$$

Algorithm 3 Computing values $Z(\alpha)$

```
1: run Algorithm 1 in both directions to get messages $\overrightarrow{W}_j(\alpha)$, $\overleftarrow{W}_i(\alpha)$. For each pattern $\alpha = ([i, j], \cdot) \in \Pi$ set
     $W(\alpha) := \frac{\overrightarrow{W}_j(\alpha)\overleftarrow{W}_i(\alpha)}{f(\alpha)}$   (19a)
     $W^-(\alpha) := \frac{\overrightarrow{W}_{j-1}(\alpha_{i:j-1})\overleftarrow{W}_{i+1}(\alpha_{i+1:j})}{f(\alpha_{i+1:j-1})}$   (19b)
2: for $\alpha \in \Pi$ (in the order of decreasing $|\alpha|$) set
     $Z(\alpha) := W(\alpha) + \sum_{\beta\in\Phi(\alpha)} [Z(\beta) - W^-(\beta)]$   (20)
```

The correctness of the algorithm is proved in (Takhanov & Kolmogorov, 2012). Let us discuss its complexity. We claim that all values $f(\alpha)$ used by the algorithm can be computed in $O(n(P + S))$ time where P and S are respectively the number of distinct non-empty prefixes and suffixes of words in Γ. Indeed, we first compute these values for patterns in the set $\overrightarrow{\Pi} \triangleq \{\alpha \mid \exists \alpha* \in \Pi^\circ\}$; by Lemma 1, this takes $O(nP)$ time. This covers values $f(\alpha)$ used in eq. (19a). As for the value in eq. (19b) for pattern $\alpha = ([i, j], \cdot) \in \Pi$, we can use the formula

$$f(\alpha_{i+1:j-1}) = \frac{f(\alpha)\tilde{c}_\alpha}{\overrightarrow{\phi}(\alpha)\overleftarrow{\phi}(\alpha)}$$

where $\tilde{c}_\alpha = c_\alpha$ if $\alpha \in \Pi^\circ$ and $\tilde{c}_\alpha = 1$ otherwise, and

$$\overrightarrow{\phi}(\alpha) = \prod_{\beta\in\Pi^\circ,\alpha=*\beta} c_\beta\ , \qquad \overleftarrow{\phi}(\alpha) = \prod_{\beta\in\Pi^\circ,\alpha=\beta*} c_\beta$$

The latter values can be computed in $O(n(P+S))$ time by applying Lemma 1 in the forward and backward directions. (In fact, there were already computed when running Algorithm 1.)

We showed that step 1 can be implemented in $O(n(P+S))$ time; let us analyze step 2. The following lemma implies that it performs $O(n\sum_{\alpha\in I(\Gamma)}|\alpha|)$ arithmetic operations; since $\sum_{\alpha\in I(\Gamma)}|\alpha| \ge \sum_{\alpha\in\Gamma}|\alpha| \ge \max\{P,S\}$, we then get that the overall complexity is $O(n\sum_{\alpha\in I(\Gamma)}|\alpha|)$.

Lemma 5. *For each $\beta\in\Pi$ there exist at most $2|\beta|$ patterns $\alpha\in\Pi$ such that $\beta\in\Phi(\alpha)$.*

Remark 1. An alternative method for computing marginals with complexity $O\left(n|\Gamma|L^2\ell^2_{\max}\right)$ was given in (Ye et al., 2009). They compute value $Z(\alpha)$ directly from messages $\overrightarrow{M}_{j'}(\cdot)$ and $\overleftarrow{M}_{i'}(\cdot)$ by summing over **pairs** of patterns (thus the square factor in the complexity). In contrast, we use a recursive rule that uses previously computed values of $Z(\cdot)$. We also use the existence of the "$\ominus$" operation. This allows us to achieve better complexity.

6. General case: $O(nP'|D|)$ algorithm

In this section and in the next one we consider the case of a general commutative semiring $(R,\oplus,\otimes)$ (without assuming the existence of an inverse operation for $\oplus$). This can be used for computing MAP in CRFs containing positive costs c_α. The algorithm closely resembles the method in (Ye et al., 2009); it is based on the same idea and has the same complexity. Our primary goal of presenting this algorithm is to motivate the $O(nP\log(\ell_{\max}+1))$ algorithm for the MAP problem given in the next section.

First, we select Π as the set of proper prefixes of patterns in Π°:

$$\Pi=\{\alpha\mid\exists\alpha+\in\Pi^\circ\}\qquad(21)$$

For each $\alpha\in\Pi_s$ we will compute message

$$M_s(\alpha)=\bigoplus_{x\in\mathcal{X}_s(\alpha;\Pi_s)} f(x)\qquad(22)$$

In order to go from step $s-1$ to s, we will use an extended set of patterns $\widehat{\Pi}_s$:

$$\begin{aligned}\widehat{\Pi}_s &= \{\alpha\mid\alpha^-\in\Pi_{s-1}\}\cup\{\varepsilon_s\} &(23)\\ &=\{\alpha c\mid\alpha\in\Pi_{s-1},c\in D^{s:s}\}\cup\{\varepsilon_s\}\end{aligned}$$

It can be checked that

$$\Pi_s\subseteq\widehat{\Pi}_s \quad\text{and}\quad \Pi^\circ_s\subseteq\widehat{\Pi}_s\qquad(24)$$

In step s we compute values $M_s(\alpha)$ in eq. (22) for all $\alpha\in\widehat{\Pi}_s$. Note, we now use the generalized definition of $\mathcal{X}_s(\alpha;\Pi_s)$ (eq. (9)) since we may have $\alpha\notin\Pi_s$. After completing step s, messages $M_s(\alpha)$ for $\alpha\in\widehat{\Pi}_s-\Pi_s$ can be discarded.

Our algorithm is given below. We have $|\widehat{\Pi}_s|=P'|D|+1$ for indexes s that are sufficiently far from the boundary, and thus the algorithm's complexity is $\Theta(nP'|D|)$ (if Lemma 1 is used for computing values $\phi(\alpha)$).

Algorithm 4 Computing $Z=\bigoplus_{x\in D^{1:n}} f(x)$

```
1: initialize messages: set $M_0(\varepsilon_0):=\mathbb{O}$
2: for each $s=1,\ldots,n$ traverse nodes $\alpha\in\widehat{\Pi}_s$ of tree
   $G[\widehat{\Pi}_s]$ starting from the leaves and set
      $M_s(\alpha):=\left[\phi(\alpha)\otimes M_{s-1}(\alpha^-)\right]\oplus\bigoplus_{(\alpha,\beta)\in E[\widehat{\Pi}_s],\beta\notin\Pi_s} M_s(\beta)$   (25)
   If $\alpha=\varepsilon_s$ then use $M_{s-1}(\alpha^-)=\mathbb{O}$
3: return $Z:=\bigoplus_{\alpha\in\Pi_n} M_n(\alpha)$
```

Theorem 6. *Algorithm 4 is correct.*

Remark 2 As we already mentioned, Algorithm 4 resembles the algorithm in (Ye et al., 2009). The latter computes the same set of messages as we do but using the following recursion: for a pattern $\alpha\in\Pi_s-\{\varepsilon_s\}$ they set

$$M_s(\alpha):=\bigoplus_{\gamma\in T_{s-1}(\alpha^-)-\bigcup_{(\alpha,\beta)\in E[\Pi_s]}T_{s-1}(\beta^-)}\phi(\gamma a)\otimes M_{s-1}(\gamma)\qquad(26)$$

where $a=\alpha_{s:s}$ is the last letter of α and $T_{s-1}(\beta)=\{\gamma\,|\,\gamma=*\beta,\gamma\in\Pi_{s-1}\}$ for $\beta\in\Pi_{s-1}$ is the set of patterns in the branch of $G[\Pi_{s-1}]$ rooted at β. It can be shown that updates (25) and (26) are equivalent: they need exactly the same number of additions (and the same number of multiplications, if $\phi(\gamma a)$ in eq. (26) is replaced with $\phi(\alpha)$ and moved before the sum).

7. General case: $O(nP\log P)$ algorithm

In the previous section we presented an algorithm for a general commutative semiring with complexity $O(nP'|D|)$. In some applications the size of the input alphabet can be very large (e.g. hundreds or thousands), so the technique may be very costly. Below we present a more complicated version with complexity $O(nP\log P)$. If $(R,\oplus,\otimes)=(\overline{\mathbb{R}},\min,+)$ then this can be reduced to $O(nP\log(\ell_{\max}+1))$ using the algorithm for *Range Minimum Queries* by (Berkman & Vishkin, 1993). We assume that $D\subseteq\Gamma$.

We will use the same definitions of sets Π_s and $\widehat{\Pi}_s$ as in the previous section, and the same intepretation of messages $M_s(\alpha)$ given by eq. (22). We need to solve the following problem: given messages $M_{s-1}(\alpha)$ for $\alpha\in\Pi_{s-1}$, compute messages $M_s(\alpha)$ for $\alpha\in\Pi_s$.

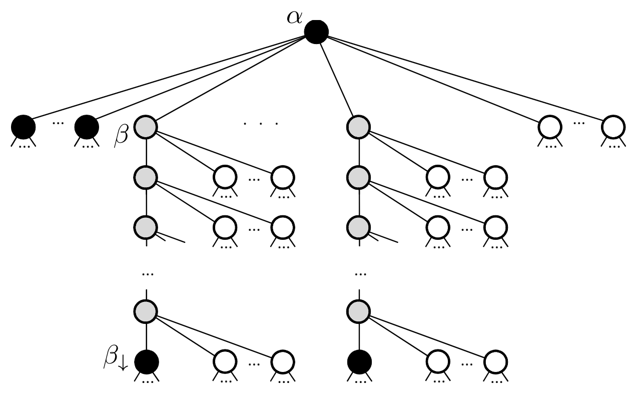

Figure 2. Structure of the subtree of $G[\widehat{\Pi}_s]$ rooted at a node $\alpha \in \Sigma_s$. White circles represent nodes in $\mathcal{A}_s$ (so all their children are also white), gray circles - nodes in $\mathcal{B}_s$, and black circles - nodes in Σ_s. Note, if $\beta \in \mathcal{B}_s$ is a child of α then $(\alpha, \beta_\downarrow) \in E[\Sigma_s]$.

Recall that in the previous section this was done by computing messages $M_s(\alpha)$ for patterns in the extended set $\widehat{\Pi}_s$ of size $O(P'|D|)$. The idea of our modification is to compute these messages only for patterns in the set Σ_s where

$$\Sigma_s^\circ \subseteq \Sigma_s \subseteq \widehat{\Pi}_s \qquad \Sigma_s^\circ = \Pi_s \cup \Pi_s^\circ \qquad |\Sigma_s| \le 2|\Sigma_s^\circ|$$

Note that $|\Sigma_s^\circ| \le P+1$. Patterns in Σ_s will be called *special.* To define them, we will use the following notation for a node $\alpha \in \widehat{\Pi}_s$:

- $\widehat{\Phi}_s(\alpha)$ is the set of children of α in the tree $G[\widehat{\Pi}_s]$.
- $\widehat{T}_s(\alpha)$ is the set of nodes in the subtree of $G[\widehat{\Pi}_s]$ rooted at α. We have $\alpha \in \widehat{T}_s(\alpha) \subseteq \widehat{\Pi}_s$.

We now define set Σ_s as follows: pattern $\alpha \in \widehat{\Pi}_s$ is special if either (i) $\alpha \in \Sigma_s^\circ$, or (ii) α has at least two children $\beta_1, \beta_2 \in \widehat{\Phi}_s(\alpha)$ such that subtree $\widehat{T}_s(\beta_i)$ for $i \in \{1,2\}$ contains a pattern from Σ_s°, i.e. $\widehat{T}_s(\beta_i) \cap \Sigma_s^\circ \neq \varnothing$.

The set of remaining patterns $\widehat{\Pi}_s - \Sigma_s$ will be split into two sets $\mathcal{A}_s$ and $\mathcal{B}_s$ as follows:

- $\mathcal{A}_s$ is the set of patterns $\alpha \in \widehat{\Pi}_s - \Sigma_s^\circ$ such that subtree $\widehat{T}_s(\alpha)$ does not contain patterns from Σ_s°.
- $\mathcal{B}_s$ is the set of patterns $\alpha \in \widehat{\Pi}_s - \Sigma_s^\circ$ such that α has exactly one child β in $G[\widehat{\Pi}_s]$ for which $\widehat{T}_s(\beta) \cap \Sigma_s^\circ \neq \varnothing$.

Clearly, $\widehat{\Pi}_s$ is a disjoint union of $\mathcal{A}_s$, $\mathcal{B}_s$ and Σ_s.

Consider a node $\alpha \in \mathcal{B}_s$. From the definition, α has exactly one link to a child in $G[\widehat{\Pi}_s]$ that belongs to $\mathcal{B}_s \cup \Sigma_s$. If this child does not belong to Σ_s, then it belongs to $\mathcal{B}_s$ and the same argument can be repeated for it. By following such links we eventually get to a node in Σ_s; the first such node will be denoted as $\alpha_\downarrow$.

We will need two more definitions. For an index t and patterns α, β ending at position t with $\beta = +\alpha$ we denote

$$W_t(\alpha) = \bigoplus_{x \in \mathcal{X}_t(\alpha)} f(x) \tag{27a}$$

$$V_t(\alpha, \beta) = \bigoplus_{x \in \mathcal{X}_t(\alpha) - \mathcal{X}_t(\beta)} f(x) \tag{27b}$$

We can now formulate the structure of the algorithm.

Algorithm 5 Computing $Z = \bigoplus_{x \in D^{1:n}} f(x)$

```
1: initialize messages: set $M_0(\varepsilon) := \mathbb{O}$
2: for each $s = 1, \dots, n$ traverse nodes $\alpha \in \Sigma_s$ of tree $G[\Sigma_s]$ starting from the leaves and set
       $$M_s(\alpha) := \phi(\alpha) \otimes [M_{s-1}(\alpha^-) \oplus A_s(\alpha) \oplus B_s(\alpha)] \oplus \bigoplus_{(\alpha,\beta) \in E[\Sigma_s], \beta \notin \Pi_s} M_s(\beta) \tag{28}$$
   where
       $$A_s(\alpha) = \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{A}_s} W_{s-1}(\beta^-) \tag{29a}$$
       $$B_s(\alpha) = \bigoplus_{\beta \in \widehat{\Phi}_s(\alpha) \cap \mathcal{B}_s} V_{s-1}(\beta^-, \beta_\downarrow{}^-) \tag{29b}$$
   If $\alpha = \varepsilon_s$ then use $M_{s-1}(\alpha^-) := \mathbb{O}$
3: return $Z := \bigoplus_{\alpha \in \Pi_n} M_n(\alpha)$
```

To fully specify the algorithm, we still need to describe how we compute quantities $A_s(\alpha)$ and $B_s(\alpha)$ defined by eq. (29a) and (29b). This is addressed by the theorem below.

Theorem 7. *(a) Algorithm 5 is correct.*
(b) There holds $|\Sigma_s| \le 2|\Sigma_s^\circ| - 1 \le 2P + 1$.
(c) Let h be the maximum depth of tree $G[\Pi_{s-1}]$. (Note, $h \le \ell_{\max} + 1$.) With an $O(P' \log h)$ preprocessing, values $V_{s-1}(\alpha, \beta)$ for any $\alpha, \beta \in \Pi_{s-1}$ with $\beta = +\alpha$ can be computed in $O(\log h)$ time.
(d) Values $A_s(\alpha)$ for all $\alpha \in \Sigma_s$ can be computed in $O(P \log P)$ time, or in $O(P)$ time when $(R, \oplus, \otimes) = (\overline{\mathbb{R}}, \min, +)$.

Clearly, the theorem implies that the algorithm can be implemented in $O(nP \log P)$ time, or in $O(nP \log(\ell_{\max}+1))$ time when $(R, \oplus, \otimes) = (\overline{\mathbb{R}}, \min, +)$. To see this, observe that the sum in (29b) is effectively over a subset of children of α in the tree $G[\Sigma_s]$ (see Fig. 2), and this tree has size $O(P)$.

8. Experiments and discussion

Experimental evaluation of pattern-based CRF on different tasks can be found in (Ye et al., 2009; Qian et al., 2009; Nguyen et al., 2011). Here we consider the problem of protein backbone dihedral angles prediction, and compare with the HMMSTR method in (Bystroff et al., 2000) which is often used as a baseline method. It is based on the HMM approach where the states of an HMM ("I-sites") are computed by clustering frequently occuring sequence-structure motifs.

We learn our pattern-based CRF using the Max-Margin approach ("struct-SVM") (Tsochantaridis et al., 2004); it calls MAP inference as a subroutine.

Experimental data was taken from the PDB database (`www.pdb.org`). The resulting sample contained 602 all-alpha proteins. Each protein in dataset is given as a pair of words: an amino-acid sequence z and a labeling x. Each amino-acid is labeled by two angles ϕ, ψ. We discretized each angle into $T = 18$ levels with the step of 20 degrees, and added a special value "unknown" (since it is sometimes present in the data); thus, the label set has size $|D| = (T+1)^2 = 361$.

We selected a set $\mathfrak{A}$ of triplets $a = (z, k, x)$ where $x \in D^k$ is a subsequence of labels, $k \in [1, |z|]$ and z is the amino-acid at the k-th position of the corresponding amino-acids subsequence.[3] We then considered the distribution

$$p(x|z) \sim \exp\{-\sum_{a\in\mathfrak{A}} w_a h_a(x,z)\} \qquad (30)$$

where $h_a(x, z)$ is the number of occurences of triplets a consistent with x and z.

We use the Hamming loss function: $\Delta(x, x') = \sum_i H(x_i, x'_i)$ where $H(\cdot,\cdot)$ is the sum of distances for angles ϕ and ψ (recall that $x_i = (\phi_i, \psi_i)$).[4] Note that struct-SVM requires the maximization of $\Delta(x, x^t) + \sum_{a\in\mathfrak{A}} w_a h_a(x, z^t)$ over labelings x; this can be done by algorithms in sections 6 or 7.

The sample set was divided into two subsets: a training sample (500) and test sample (102). The training sample was used to train parameters for different regularization parameters C of struct-SVM and subsequent choice of the best one, and the resulting prediction accuracy was assessed on the test set. As a measure of accuracy, we used the **MDA** value from (Bystroff et al., 2000), which is the fraction of residues in segments of length 8 with no greater than 120 maximum deviation between observed and predicted backbone torsion angles. We obtained the result **MDA** = 59.0%. This approximately corresponds to the result of HMMSTR (57.1% on a training data of 618 randomly selected proteins, and 59.1% on a test data of size 61). However, they used an additional trick, namely clustering frequently occuring sequence-structure motifs. This allowed them, in particular, to utilize more information, for example, the whole sequence of the amino-acids for a given motif was taken into account (while we look only at the first and the last amino-acids of a pattern).

[3] We took triplets that occur more then 20 times in the database; this gave 7467 triplets. The average pattern length was 3.4, with 6 being the maximum.

[4] For discretized angles ϕ we use $H_\phi(\phi, \phi') = \min\{|\phi - \phi'|, T - |\phi - \phi'|\}$. If one of the labels is "unknown" and the other one is not then we use $H_\phi(\phi, \phi') = T$ instead. The distance for angles ψ is defined similarly.

We thus expect that with a better "feature selection" step the results of pattern-based CRFs can be improved. In (Ye et al., 2009) this was done by boosting type methods (Dietterich et al., 2004) with sequential selection of useful features. Another improvement can be made by considering "superpatterns" by which we mean clusters of patterns (where patterns are not only sequences of labels, but also different local structures like "triplets" that we used) that have the same weights in learning model. In this case a problem of clusterization of patterns set is to be addressed before learning of weights. It is also important to note that our algorithms can adapt to formulation with long superpatterns, since their complexities depend linearly on the number of patterns.

A C++ implementation of our algorithms can be found at `http://pub.ist.ac.at/~vnk/software.html`.

9. Conclusions and future work

The HMM/CRF models are a method of choice for many sequence labeling problems. Pattern-based CRFs generalize these models by providing more flexibility; they allow efficiently encoding certain long-range interactions. We provided a complete set of tools for inference in pattern-based CRFs. Importantly, all of our algorithms are linear in the number of patterns.

Our preliminary experiments show a potential of this model for the protein dihedral angles prediction problem. With a rather basic model were able to obtain results close to that in (Bystroff et al., 2000) who used many optimizations. To improve results further, we believe it is important to introduce techniques for parameterizing pattern-based CRFs, including efficient algorithms for feature selection and superpatterns clusterization. This will be our future work.

Acknowledgements We thank Herbert Edelsbrunner for helpful discussions.

References

Berkman, Omer and Vishkin, Uzi. Recursive star-tree parallel data structure. *SIAM Journal on Comput-*

ing, 22(2):221–242, 1993.

Bystroff, C., Thorsson, V., and Baker, D. HMMSTR: a hidden Markov model for local sequence-structure correlation in proteins. *J Mol. Biol.*, 301:173–190, 2000.

Dietterich, T. G., Ashenfelter, A., and Bulatov, Y. Training conditional random fields via gradient tree boosting. In *ICML*, 2004.

Komodakis, N. and Paragios, N. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.

Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

Nguyen, Viet Cuong, Ye, Nan, Lee, Wee Sun, and Chieu, Hai Leong. Semi-Markov conditional random field with high-order features. In *ICML 2011 Structured Sparsity: Learning and Inference Workshop*, 2011.

Qian, Xian, Jiang, Xiaoqian, Zhang, Qi, Huang, Xuanjing, and Wu, Lide. Sparse higher order conditional random fields for improved sequence labeling. In *ICML*, 2009.

Rother, C., Kohli, P., Feng, W., and Jia, J. Minimizing sparse higher order energy functions of discrete variables. In *CVPR*, 2009.

Sarawagi, S. and Cohen, W. Semi-Markov conditional random fields for information extraction. In *NIPS*, 2004.

Takhanov, R. and Kolmogorov, V. Inference algorithms for pattern-based CRFs on sequence data. *ArXiv*, abs/1210.0508, 2012.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Vose, M. D. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.

Ye, Nan, Lee, Wee Sun, Chieu, Hai Leong, and Wu, Dan. Conditional random fields with high-order features for sequence labeling. In *NIPS*, 2009.