ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Factored sequence kernels

Pierre Mahé and Nicola Cancedda [*]

Xerox Research Centre Europe
6, chemin de Maupertuis, 38240 Meylan - France

Abstract. In this paper we propose an extension of sequence kernels to the case where the symbols that define the sequences have multiple representations. This configuration occurs in natural language processing for instance, where words can be characterized according to different linguistic dimensions. The core of our contribution is to integrate early the different representations in the kernel, in a way that generates rich composite features defined across the various symbol dimensions.

1 Introduction

Recent years saw important developments in the field of kernel methods for complex data with, in particular, the advent of kernels for structured objects such as sequences, trees or graphs on the one hand, and advanced methods to combine multiple and heterogeneous sources of information in a global kernel function on the other hand. Combined with the family of so-called kernel methods, this offers a versatile and powerful framework to apply machine learning techniques to real-world problems [1]. In this paper we propose an extension of sequence kernels [2] to the case where the symbols that define the sequences have multiple representations. This configuration occurs in natural language processing for instance, where words can be characterized according to different linguistic dimensions, often called *factors* [3, 4]. The novelty of our approach is to integrate early the different factors in the kernel, in a way that generates rich composite features defined across the various symbol representations. The paper is structured as follows. Section 2 briefly introduces sequence kernels and details our kernel construction, Section 3 validates the approach on an artificial problem, and Section 4 discusses future work.

2 Factored kernel

In this section, we introduce sequence kernels, define the notion of factored representation, and detail our kernel formulation.

2.1 Sequence kernels

Let x be a sequence of $|x|$ symbols drawn from an alphabet Σ. A subsequence s_x of length n is identified by a set of n symbol indices constrained to be strictly increasing: $s_x = x[\mathbf{i}]$, with $\mathbf{i} \in \{1, |x|\}^n$ and $\mathbf{i}[1] < \mathbf{i}[2] < \cdots < \mathbf{i}[n]$. We let $\mathcal{S}_n(x)$

[*]This work was supported by the IST Programme of the European Community under the SMART project, IST-2005-033917, and under the PASCAL Network of Excellence, IST-2002-506778.

be the set of subsequences of size n associated to x, and we denote as $g(s_x)$ the number of gaps contained in the subsequence s_x[1]. With this notation at hand, we can state the following definition, upon which is based the rest of the paper.

Definition 1. *For the pair of sequences x and y over an alphabet Σ, the* gap-weighted subsequence kernel *of order n is defined as $k_n(x, y) = \langle\phi(x), \phi(y)\rangle$, where:*

- $\phi(x) = \big(\phi_u(x)\big)_{u\in\Sigma^n}$
- $\phi_u(x) = \sum_{s_x\in\mathcal{S}_n(x)} \mathbf{1}(s_x = u)\lambda^{g(s_x)}$

The parameter λ is called the gap penalization factor. When it tends to zero, any subsequence containing gaps gets so down-weighted that the kernel boils down to counting the number of contiguous subsequences. When it is increased, the penalization of gaps decreases, and for $\lambda = 1$, every subsequence contributes equally to the kernel value, whatever its number of gaps is. This kernel was initially introduced to compare sequences of characters [2] and was later extended to compare sequences of words [5].

2.2 Factored representation and kernels combination

We now consider the case where the symbols that define the sequences can be represented along several dimensions, that we call *factors* in the following. Under this representation, a symbol u is formally defined as a tuple of p factors $\{u^{(d)}\}_{d=1:p}$ drawn each from a different alphabet Σ_d. A simple way to define a kernel k on such multi-dimensional sequences is to resort to a linear combination of individual kernels [6, 7]:

$$k(x, y) = \sum_{d=1}^{p} w_d k_n(x^{(d)}, y^{(d)}), \tag{1}$$

where $x^{(d)}$ denotes the d-th sequence of symbol factors associated to the sequence x, k_n is the kernel[2] of Definition 1 and the weights $w_d \geq 0$ control the relative contribution of the different factors in the global kernel. It is easy to see that the kernel of Equation (1) can be written as a standard inner product $k(x, y) = \langle\phi(x), \phi(y)\rangle$, with:

- $\phi(x) = \big(\phi_u(x)\big)_{u\in\Sigma}$, with $\Sigma = \bigcup_{d=1}^{p}(\Sigma_d)^n$,
- for a sequence $u \in (\Sigma_d)^n$, $\phi_u(x) = \sqrt{w_d}\sum_{s_x\in\mathcal{S}_n(x^{(d)})} \mathbf{1}(s_x = u)\lambda^{g(s_x)}$.

In other words, the linear combination of kernels has the basic effect of concatenating their associated feature spaces, which boils down in our case to representing a sequence by its subsequences that can be extracted along the different

[1]If $s_x = x[\mathbf{i}] \in \mathcal{S}_n(x)$, we have $g(s_x) = \mathbf{i}[n] - \mathbf{i}[1] - n + 1$.

[2]For simplicity, we introduce a single kernel order n to compare the different sequences, but one could use factor-specific orders $n(d)$.

dimensions. Each dimension of the feature space is the (gap-penalized) number of corresponding subsequences, multiplied by a scaling factor reflecting the introduction of weights in the linear combination.

2.3 Factored kernel definition

It is well known that the kernel of Definition 1 can equivalently be written in the context of convolution kernels [8] as:

$$\begin{aligned} k_n(x,y) &= \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \mathbf{1}(s_x = s_y) \lambda^{g(s_x)} \lambda^{g(s_y)} \\ &= \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} \mathbf{1}(s_x[i] = s_y[i]). \end{aligned}$$

Under this form, the kernel definition can considerably be enriched by the introduction of an arbitrary kernel k_Σ between sequence symbols:

$$k(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} k_\Sigma(s_x[i], s_y[i]), \tag{2}$$

This is known as the *soft-matching* extension of sequence kernels, and is known to be valid provided the kernel k_Σ is a proper kernel function [1]. To accommodate factored representations, we propose to resort to this soft-matching formulation, according to the following definition.

Definition 2. *The* factored kernel *is defined as a soft-matching sequence kernel (2), where for the pair of symbols (u,v) the kernel k_Σ is defined as:*

$$k_\Sigma(u,v) = \sum_{d=1}^{p} w_d \mathbf{1}(u^{(d)} = v^{(d)}).$$

Soft-matching sequence kernels do not have an explicit interpretation in terms of inner products and feature spaces. In our case however, we can state the following proposition, whose proof is post-poned to the appendix.

Proposition 1. *The factored kernel of Definition 2 can be written as a standard inner product $k(x,y) = \langle \phi(x), \phi(y) \rangle$, where:*

- $\phi(x) = \big(\phi_u(x)\big)_{u \in \Sigma}$, *with* $\Sigma = \big(\bigcup_{d=1}^{p} \Sigma_d\big)^n$,
- *for the (composite) sequence* $u \in \Sigma_{f_1(u)} \times \cdots \times \Sigma_{f_n(u)}$, $f_i(u) \in \{1,\ldots,p\}$, *we have* $\phi_u(x) = w(u) \sum_{s_x \in \mathcal{S}_n^*(x)} \mathbf{1}(s_x = u)\lambda^{g(s_x)}$, *where* $w(u) = \prod_{i=1}^{n} \sqrt{w_{f_i(u)}}$ *and* $\mathcal{S}_n^*(x)$ *denotes the set of composite subsequences of* x, *defined as:*

$$\mathcal{S}_n^*(x) = \big\{ x^{(\mathbf{d})}[\mathbf{i}] = x^{(d_1)}[i_i] \ldots x^{(d_n)}[i_n], \\ \mathbf{i} \in \{1,\ldots,|x|\}^n, \mathbf{d} \in \{1,\ldots,p\}^n, \mathbf{i}[1] < \mathbf{i}[2] < \cdots < \mathbf{i}[n] \big\}.$$

In comparison with the linear combination of kernels, the key feature of this factored integration is therefore the ability to detect composite subsequences in which symbols are drawn from the union of the individual alphabets $\Sigma = \bigcup_{d=1}^{p} \Sigma_d$. Intuitively, the factored kernel can be seen as i) merging the individual alphabets into a global alphabet, and ii) counting the resulting composite subsequences in the factored representation. On the other hand, linearly combining the kernels corresponds to i) counting subsequences in the different dimensions, and ii) merging the resulting feature vectors. In both cases, albeit in a different way, the weight parameters w_d allow to control the relative contribution of the different factors in the feature space. As an interesting by-product, the factored integration can be seen to be slightly more efficient from the computational point of view. Indeed, while the linear combination of kernel basically multiplies the initial complexity by a factor p, leading to an overall complexity of $O(np|x|y|)$, the soft-matching extension has the same $O(n|x|y|)$ complexity as the hard-matching one, provided the symbol similarity measure is computed in advance. The complete k_Σ Gram matrix would in our case be of size $(|\Sigma_1||\Sigma_2|\dots|\Sigma_p|)^2$, and in many cases it would be highly impractical to pre-compute it and to store it in memory. Computing it "on the fly" adds an extra step of $O(p|x||y|)$ to the kernel evaluation, leading to an overall complexity of $O\big((n+p)|x||y|\big)$.

3 Experimental results

To validate the kernel construction, we consider a simple task aiming to distinguish "good" and "bad" word sequences [5], where "good" sequences are typical sentences, and "bad" sequences are disrupted sentences, in which pairs of consecutive words are swapped with probability 0.16. We use a tridimensional factored representation where a word is represented by its surface form, its lemma and its part of speech (POS). Albeit artificial, this problem can be seen to be relevant to the field of Statistical Machine Translation where word misorderings are important to detect. We consider training and test sets of respectively 5000 and 2000 sentences extracted from the Europarl corpus[3] and use the python package PyML[4] to perform SVM classification. The soft margin parameter of the SVM is systematically optimized by cross validation on the training set, and we report classification accuracy on the test set.

Table 1 shows the result obtained by kernels computed independently on the different representations of the sentences, for varying n and λ. For lack of space we do not go into a detailed analysis of these results, but we note that i) optimal results are systematically obtained for $n = 2$ and $\lambda = 0.001$, and ii) POS significantly outperform surface forms and lemmas. Table 2 show the results obtained by the linear combination of individual kernels (1st line) and the factored kernel (2nd line) for different weight vectors. In both cases, $n = 2$ and $\lambda = 0.001$, but the weights used in the factored kernel are set to the square root of the weights used in the linear combination. This configuration aims to give an identical

[3]`http://www.statmt.org/europarl`
[4]`http://pyml.sourceforge.net`

	$n = 2$				$n = 3$				$n = 4$			
λ	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0	0.001	0.1	0.5	1.0
surface	**85.1**	84.3	80.1	56.1	73.0	73.1	72.0	58.1	65.1	64.5	63.7	57.0
lemma	**85.6**	85.0	79.3	56.3	73.9	73.6	73.0	58.5	65.1	64.7	64.7	57.6
pos	**89.7**	89.6	85.1	63.9	88.1	88.3	87.8	69.3	85.5	86.7	87.5	74.5

Table 1: Test set accuracy obtained with individual word-sequence kernels.

weight vector **w**	[1 1 1]	[1 1 2]	[1 1 5]	[1 1 10]	[1 2 5]
combined kernel (1)	89.9	90.3	90.9	**91.2**	90.3
factored kernel (2)	91.6	91.6	**92.1**	91.9	91.6

Table 2: Test set accuracy obtained by the linear combination of kernels (1) and the factored kernel (2), $\mathbf{w} = [w_{\text{surf}} \;\; w_{\text{lem}} \;\; w_{\text{pos}}]$.

weight to the features that are present in both configurations: the subsequences drawn from a single alphabet[5]. For lack of space, we simply note that i) linearly combining individual kernel indeed improves the best result that can be obtained on separated factors, and ii) the factored integration further improves over the linearly combined kernel. These preliminary experiments therefore account for the benefit of being able to consider composite subsequences.

4 Conclusion

In this paper, we have presented a simple kernel between multi-dimensional sequences having the key feature of defining composite subsequences. Factored representations are gaining popularity in the field of natural language processing, where they offer the possibility to introduce grammatical information in language models [3] and statistical machine translation systems [4] for instance. By their ability to leverage linguistic resources in a global and systematic way, we believe that factored kernels can constitute powerful tools in this context, and we are now working toward their integration in (discriminative) language models. Another direction we are taking consists in extending this framework to the integration of linguistic ressources having a higher level of complexity, such as morphological and semantic features.

Appendix - proof of Proposition 1

We detail the feature space interpretation of the factored kernel:

$$k(x, y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \prod_{i=1}^{n} \sum_{d=1}^{p} w_d \mathbf{1}(s_x^{(d)}[i] = s_y^{(d)}[i]).$$

[5]Recall from Section 2 that sequences drawn from Σ_d are given a weight of $\sqrt{w_d}$ in the linear combination of kernels, and $\prod_{i=1}^{n} \sqrt{w_d}$ in the factored kernel.

A simple distributivity argument allows to permute the d-summation and the i-product, by means of a vector of factor indices $\mathbf{d} = [d_1, \ldots, d_n] \in \{1, p\}^n$:

$$k(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \sum_{\mathbf{d}} \prod_{i=1}^{n} w_{d_i} \mathbf{1}(s_x^{(d_i)}[i] = s_y^{(d_i)}[i]).$$

The product of binary functions between individual word factors can then be replaced by a binary function between sequences of word factors:

$$k(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)+g(s_y)} \sum_{\mathbf{d}} w(\mathbf{d}) \mathbf{1}(s_x^{(\mathbf{d})} = s_y^{(\mathbf{d})}),$$

with $w(\mathbf{d}) = \prod_{d=1}^{n} w_{d_i}$. Using the notion of *composite subsequence* introduced in Proposition 1, the summation over the vector $\mathbf{d}$ of word factors can be included in the summation over pairs of subsequences:

$$k(x,y) = \sum_{s_x \in \mathcal{S}_n^*(x)} \sum_{s_y \in \mathcal{S}_n^*(y)} \lambda^{g(s_x)+g(s_y)} w(s_x) \mathbf{1}(s_x = s_y), \tag{3}$$

where for $s_x = x^{(\mathbf{d})}[\mathbf{i}] \in \mathcal{S}_n^*(x)$ and $s_y = y^{(\mathbf{d}')}[\mathbf{i}'] \in \mathcal{S}_n^*(y)$, $\mathbf{1}(s_x = s_y) = 1$ iff $\mathbf{d} = \mathbf{d}'$ and $x^{(d_k)}[i_k] = y^{(d_k)}[i'_k]$ for $k = 1, \ldots, n$; and $w(s_x) = \prod_{i=1}^{n} w_{d_i}$. Finally, because $\mathbf{1}(s_x = s_y) = 1$ implies $w(s_x) = w(s_y)$, the kernel of Equation (3) can be written as a standard inner product $k(x,y) = \langle \phi(x), \phi(y) \rangle$, where $\phi(x) = (\phi_u(x))_{u \in \Sigma}$ with $\Sigma = \left(\bigcup_{d=1}^{p} \Sigma_d\right)^n$, the set of all possible composite sequences of length n, and $\phi_u(x) = \sum_{s_x \in \mathcal{S}_n^*(x)} \mathbf{1}(s_x = u) \lambda^{g(s_x)} \sqrt{w(s_x)}$.

The proof is concluded by noting that for $u \in \prod_{i=1}^{n} \Sigma_{f_i(u)}$ and $s_x = x^{(\mathbf{d})}[\mathbf{i}]$, the equality $\mathbf{1}(s_x = u) = 1$ implies $d_i = f_i(u)$ for $i = 1, \ldots, n$, and therefore $w(s_x) = \prod_{i=1}^{n} w_{d_i} = \prod_{i=1}^{n} w_{f_i(u)}$.

References

[1] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[2] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *J. Mach. Learn. Res.*, 2:419–444, 2002.

[3] J. A. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *NAACL '03*, pages 4–6, 2003.

[4] P. Koehn and H. Hoang. Factored translation model. In *EMNLP'07*, pages 868–876, 2007.

[5] N. Cancedda, E. Gaussier, C. Goutte, and J.-M. Renders. Word-sequence kernels. *J. Mach. Learn. Res.*, 3:1059–1082, 2003.

[6] G.R.G Lanckriet, N. Cristianini, M.I Jordan, and W.S Noble. Kernel-based integration of genomic data using semidefinite programming. In B. Schölkopf, K. Tsuda, and J.P. Vert, editors, *Kernel Methods in Computational Biology*, pages 231–259. MIT Press, 2004.

[7] C. Giuliano, A. Gliozzo, and C. Strapparava. Syntagmatic kernels: a word sense disambiguation case study. In *EACL'06 - Workshop on Learning Structured Information in Natural Language Applications*, 2006.

[8] D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.