A Generative Multiset Kernel for Structured Data

Davide Bacciu[1], Alessio Micheli[1], and Alessandro Sperduti[2]

[1] Dipartimento di Informatica, Università di Pisa,
`{bacciu,micheli}@di.unipi.it`
[2] Dipartimento di Matematica, Università di Padova,
`sperduti@math.unipd.it`

Abstract. The paper introduces a novel approach for defining efficient generative kernels for structured-data based on the concept of multisets and Jaccard similarity. The multiset feature-space allows to enhance the adaptive kernel with syntactic information on structure matching. The proposed approach is validated using an input-driven hidden Markov model for trees as generative model, but it is enough general to be straightforwardly applicable to any probabilistic latent variable model. The experimental evaluation shows that the proposed Jaccard kernel has a superior classification performance with respect to the Fisher Kernel, while consistently reducing the computational requirements.

1 Introduction

Generative kernels exploit the knowledge captured by a probabilistic model to define a similarity metrics for complex data, including structured information such as sequences, trees and graphs. The probabilistic model is, first, trained to learn the generating (parametric) distribution of input data. Then, the kernel determines the similarity of two samples by comparing how well they are generated (*explained*) by the probabilistic model. In practice, these approaches typically represent the input information in a feature space generated by the parameters (or by the distribution) of the generative model. This makes the generative kernel approach quite general, as it can be applied to any problem for which a suitable probabilistic model can be devised and fitted. However, the parameter space of a probabilistic model quickly grows with the complexity of the problem, e.g. the number of vertices in a graph or the categories in a classification task, making kernel computation potentially cumbersome. This is the case, for instance, of the Fisher kernel [1], whose representation founds on the derivative of the likelihood of input data with respect to the model parameters. Additionally, these approaches tend to rely only on the discriminative power of the underlying generative model while, when dealing with structured-data, it might be necessary to provide the kernel with some insight on how structures should be compared, i.e. which structural agreements are more relevant. Syntactical kernels [2] focus on this latter aspect, exploiting apriori knowledge to determine how structure similarity should be measured. Generative kernels, on the other hand, learn the similarity metric adaptively by observing the data.

In this paper, we propose a novel hybrid approach that adaptively learns the kernel from an input-driven generative model for trees [3], while allowing to inject syntactical information into the induced feature space. The approach founds on a novel feature space representation where inputs are transformed into multisets, i.e. vectors of counts, and are compared through Jaccard similarity [4]. The multiset provides information on the latent variables of the probabilistic model that are responsible for generating the input data. For instance, given an input sequence, the corresponding multiset would represent the counts of hidden states activated in the generating Hidden Markov Model (HMM). The proposed model is quite general, as it can be applied to any probabilistic model based on categorical latent variables. Further, by virtue of the compact multiset representation, it can consistently reduce the computational complexity of kernel calculation with respect to the Fisher kernel.

2 Generative Kernels for Tree-Structured Data

In Section 2.1, we summarize the key aspects of the Fisher Kernal approach, i.e. the reference generative kernel for tree-structured data. In Section 2.2 we propose a novel class of efficient generative kernels for categorical latent variable model, that exploits the Jaccard multiset similarity.

2.1 The Fisher Kernel for Trees

The term Fisher kernel [1] denotes a general class of kernels that can be derived out of any parametric generative model. The underlying idea of the approach is to represent an input sample x in a feature space defined by the derivative of the log-likelihood $\log P(x|\theta)$ of the generative model, with respect to its parameters θ. Such a derivative, known as the *Fisher score* vector $U_x = \nabla_\theta \log P(x|\theta)$, can be used to define the Fisher kernel $k(x_1, x_2) = U_{x_1}^T I^{-1} U_{x_2}$, where $I = \mathbb{E}_x[U_x^T U_x]$ is the Fisher Information matrix and $\mathbb{E}_x$ is the expectation over x. In practice, the Fisher kernel is often approximated by the *practical* kernel $k(x_1, x_2) = \langle U_{x_1}, U_{x_2} \rangle$, given the minor significance of the information matrix contribution. An element of the Fisher score vector represents the contribution of the corresponding model parameter to the generation of the input sample: hence, the Fisher kernel measures the similarity of two input samples in terms of their distance in the feature space generated by the model parameters.

The Fisher kernel has been introduced in [1] with application to sequential data classification, using the HMM as generative model for the sequences. This has later been extended to deal with tree-structured data [5], in particular in phylogenetic trees [6], using the Hidden Tree Markov Model (HTMM) [7] as a generative model for the structured samples. The Fisher score of a (multinomial) HTMM is computed by differentiating its log-likelihood with respect to the state transition matrix $A_{ij} \in \mathbb{R}^{C \times C}$, the prior probability $\pi_i \in \mathbb{R}^C$ and the emission matrix $b_i(o) \in \mathbb{R}^{C \times M}$, where C is the number of hidden Markov states and M is the size of the finite node labels o (results can be generalized to continuous node labels with a Gaussian emission). This is calculated, in practice, using the sufficient statistics obtained by the *upwards-downwards* algorithm [7], that is a dynamic programming algorithm that efficiently computes the E-step of the HTMM learning algorithm, by exploiting message passing over the structure of the observed tree.

The size of the Fisher feature space depends on the number of hidden Markov states C, as well as on the stationarity assumption that are taken. A reasonable assumption is to consider positional models, i.e. where the state transition matrix $A_{ij}(l)$ is dependent on the position l of a node among its siblings. Given an input tree $\mathbf{x}_n$ with a variable number of nodes N_n, this is transformed to a Fisher score vector $\Phi(\mathbf{y}_n)$ of fixed length $C^2 \cdot L + C + M \cdot C$, where L is the maximum number of non-empty children in the tree dataset. In a multi-class classification task, it is needed to train a separate HTMM for each of the V classes, resulting in a total size of the Fisher feature space of $(C^2 \cdot L + C + M \cdot C) \cdot V$. In order to compute the Fisher score representation $\Phi(\mathbf{x}_n)$ of the tree, we need to run the upwards-downwards algorithm on $\mathbf{y}_n$ for each of the V HTMMs, with a cost that is $O(2U_n \cdot C \cdot L \cdot V)$, where U_n is the number of nodes that varies from tree to tree. The cost for computing the practical Fisher kernel $k(\mathbf{x}_1, \mathbf{x}_2)$ from V trained HTMMs is, then, $O\left((C^2 \cdot L + C + M \cdot C) \cdot V\right)$ for each couple of trees.

The Fisher kernel for trees [5] is not the sole generative kernel for structured data that exploits the HTMM parameter space to define a suitable kernel embedding: [6] discusses an alternative feature space obtained by concatenating the sufficient upwards-downwards statistics of an HTMM, as well as a tree kernel based on the probability product approach by Jebara et al [8]. A comparative analysis [6] shows that the Fisher Kernel has the best performance among the three in tree classification tasks. Therefore, in the following, we use the Fisher kernel as a reference model for assessing the experimental performance of the newly-proposed Jaccard-based approach.

2.2 A General Jaccard Kernel for Hidden State Multisets

The Fisher approach yields to discriminative kernel matrices but requires a high-dimensional embedding that can become computationally impractical for large datasets and high-dimensional trees. In the following, we describe an efficient generative kernel for trees that defines a small, yet discriminative, feature space founding on the concept of multiset similarity.

The building block of the kernel is again a generative hidden Markov model for tree-structured data. In particular, we build on the *Input-Output Bottom-up Hidden Tree Markov Model* (IO-BHTMM) [3], a recently proposed input-driven generative model for tree-structured data. This choice is motivated by the need of defining a truly adaptive kernel, whose metric, induced by the underlying probabilistic model, explicitly takes into account the discriminative information of the target classes. We expect that, by using a more sophisticated probabilistic model, we can reduce the kernel complexity while maintaining, or even increasing its discriminative performance.

The homogenous generative approach, used for instance by HTMM, requires training a separate probabilistic model for each class and then categorizing a test sample based on the model that has the highest likelihood. During training, each probabilistic model sees only positive examples from its target class, hence it cannot easily determine which *piece* of input information is truly discriminative, i.e. it differentiates the samples in the target class from those in other categories. As a result, they may be confused by non-discriminative substructures that recur in different classes. Conversely, the input-driven approach of IO-BHTMM allows training a unique generative model that learns a transduction from the input tree to a target structure which, in the simplest case, represents the sample target class. Homogenous approaches learn the unconditional model $P(\mathbf{x}_n|\theta)$, where the input trees $\mathbf{x}_n$ are the outcome of the generative process parameterized by θ. The input-driven approach, on the other hand, learns the input-conditional distribution $P(\mathbf{y}_n|\mathbf{x}_n, \theta)$, where the input trees $\mathbf{x}_n$ condition the generative process of an output structure $\mathbf{y}_n$ that, e.g., represents the target information. Hence, in a classification setting, the IO-BHTMM is capable of learning to correlate the structural input information $\mathbf{x}_n$ with the target classes, identifying discriminative portions of the input structures and providing the generative kernel with a truly adaptive metric.

As in standard HMMs, IO-BHTMM learns the conditional generative model $P(\mathbf{y}^n|\mathbf{x}^n, \theta)$ by exploiting multinomial hidden state variables Q_u following the same indexing as the observed nodes u, with values over the finite set $[1, \dots, C]$. The introduction of the hidden state Q_u allows to simplify the conditional probabilistic model, but can also be exploited to define a generative kernel for trees based on the notion of *multisets*. Given a trained IO-BTHMM, we can transform an input tree $\mathbf{x}_n$ into a *bag-of-states*, that is a vector of hidden state counts, similarly to how textual documents are represented as vectors of word counts. By using the *Viterbi Algorithm* (see [3] for details), it is possible to determine the most likely hidden state assignment $Q^*_{n,u}$ for each node u in an input tree $\mathbf{x}_n$. Using the Viterbi states it is possible to define several bag-of-states encodings, depending on the amount of syntactical, i.e. structural, information that we want to introduce in the kernel feature-space representation.

The simplest form of multiset is the *unigram*, where an input tree $\mathbf{x}_n$ is transformed to a C-dimensional feature-vector $\Phi(\mathbf{x}_n)$ such that its i-th component is

$$\Phi_i(\mathbf{x}_n) = \sum_{u \in U_n} \delta(Q^*_{n,u}, i) \quad \text{and} \quad i = 1, \dots, C$$

where U_n is the set of nodes in the n-th tree and $\delta(\cdot, \cdot)$ is the Kronecker function. Such a feature-space representation captures information on the prevalent *topics* in the tree, but does not convey any structural information, besides that captured by the generative model and conveyed by the hidden state assignment. To introduce some form of syntactical knowledge, we define a *bigram* representation, such that an input tree $\mathbf{x}_n$ is transformed to a (C^2)-dimensional feature-vector

$\Phi(\mathbf{x}_n)$ such that its i_j element is

$$\Phi_{i_j}(\mathbf{x}_n) = \sum_{u \in U_n} \sum_{l \in ch(u)} \delta(Q^*_{n,u}, i)\delta(Q^*_{n,l}, j) \text{ and } j = 1, \ldots, C, \; i_j = 1, \ldots, C$$

where $ch(u)$ is the set of children of node u. The bigram representation allows to represent the co-occurrence of hidden states patterns between a parent node and each of its children taken independently, thus providing the kernel with some form of structural information. The proposed approach is general and can be taken further by introducing increasing amounts of syntax in the feature space (e.g. by considering the $(L + 1)$-gram of a node jointly with its L children), at the cost of an increase in the feature space size. For the purpose of this paper, we limit our analysis to the unigram and the bigram representation, and to a combination of the two obtained by concatenating the unigram with the bigram into a $(C + C^2)$-dimensional feature space (*unibigram* in the following).

Once chosen a multiset representation for the trees, we need to define an appropriate kernel for such a feature space. We propose the *Jaccard similarity* [4], that is a well known metric for comparing multisets and that, in its most general form, writes as

$$J(Z_1, Z_2) = \frac{f(Z_1 \cap Z_2)}{f(Z_1 \cup Z_2)} \qquad (1)$$

where f is a suitable function (e.g. cardinality). For the purpose of this paper, we define the *Jaccard kernel* for trees as the multiset Jaccard similarity

$$k(\mathbf{x}_1, \mathbf{x}_2) = \frac{\sum_{i=1}^{D} \min(\Phi_i(\mathbf{x}_1), \Phi_i(\mathbf{x}_2))}{\sum_{i=1}^{D} \max(\Phi_i(\mathbf{x}_1), \Phi_i(\mathbf{x}_2))} \qquad (2)$$

where $\Phi(\dot{})$ is one of the multiset encodings described above and D its the corresponding size of the feature space. For instance, the Jaccard kernel for IOB-HTMM unibigram can be computed in $O(D) = O(C^2 + C)$, that is computationally more efficient than the corresponding Fisher kernel, especially when dealing with tasks with a non-trivial number of classes V and a large input vocabulary M. In this paper, we focus on evaluating the efficacy and efficiency of a simple Jaccard kernel exploiting the adaptive information of an input-conditional IO-BHTMM. Nevertheless, the proposed Jaccard approach is general and can be seamlessly applied to different generative models and data-types.

3 Experimental Evaluation

Experiments on two data sets from the 2005 and 2006 INEX Competition [9] have been performed to benchmark the proposed Jaccard kernel against the Fisher kernel for trees on a classification task. INEX 2005 comprises 9,361 trees, 11 classes, with 366 possible node labels. INEX 2006 comprises 12,107 trees, 18 classes, and 65 possible labels. Standard training and test sets are available for both datasets [9], with a 50%-50% split. IO-BHTMM has been trained to

Table 1. Jaccard Kernel results on the INEX05 and INEX06 datasets: mean test set error is shown (standard deviation is in brackets). For each kernel type, D is the size of the feature space.

	INEX 2005						INEX 2006	
	Unigram		Bigram		Unibigram		Bigram	Unibigram
	D	Error	D	Error	D	Error	Error	Error
C = 8	8	6.61 (0.48)	64	5.11 (0.47)	72	4.88 (0.17)	71.25 (0.78)	72.87 (1.5)
C = 10	10	6.26 (0.58)	100	4.97 (0.15)	110	4.75 (0.26)	72.09 (0.74)	72.25 (0.37)
C = 16	16	6.75 (0.55)	256	5.70 (0.49)	272	5.20 (0.17)	72.02 (0.71)	72.87 (1.13)

transform an input INEX tree into an isomorphic output tree having the corresponding class label on each of its nodes. The large number of classes in the datasets makes them challenging benchmarks, such that the random classifier baseline for INEX 2005 and INEX 2006 is 9% and 5.5%, respectively. The computation of both the Fisher and the Jaccard kernel requires, first, to train the underlying generative models, that are HTMM and IO-BHTMM, respectively. The emission distribution of the generative models has been set to a multinomial for both datasets. Both generative models have been trained on the standard training sets, with different configurations (i.e. number of hidden states) and varying initial conditions. A different HTMM has been trained for each class, while varying the number of hidden states in $\{6, 8, 10\}$ for each trial. Conversely, for each trial, a single IO-BHTMM has been trained to transform the input INEX trees into isomorphic output trees having the corresponding class label on each of the nodes. Again, several IO-BHTMM configurations have been tested by varying the number of hidden states in $\{8, 10, 16\}$. To account for the fluctuations in performance due to different initial conditions, we have generated 5 model repetitions for each configuration. In both approaches, the hidden state assignment for the test trees has been obtained through Viterbi maximization [7, 3]. The Gram matrices generated by the Fisher kernel could not be used straightforwardly, since the small posterior probabilities in the denominator of the kernel equations [1] tend to produce dense kernel matrices, whose elements have artificially high values even for dissimilar samples. As noted by [1], a suitable solution (also adopted in this paper) is to normalize the Fisher kernel. This requires an additional $O(N^2)$ computation, where N is the dataset size.

SVM-based multiclass classification has been produced by LIBSVM [10] (C-SVM classifier), using the gram-matrices obtained from the Jaccard and Fisher kernels. For each kernel matrix, the cost parameter of the C-SVM was selected on the validation set, using a 3-fold cross validation scheme, from the following values: 0.001, 0.01, 0.1, 1, 10, 100, 1000. The final SVM, with the cost value selected on the validation set, was trained on the union of the training and validation sets and evaluated on the test set. Table 1 and 2 show the classification error on the test set of the Jaccard and Fisher kernels, respectively, averaged over the 5 repetitions, as a function of the feature space size D. The results on the INEX 2005 dataset show that the Jaccard kernel yields to a better classifi-

Table 2. Fisher Kernel results on the INEX05 and INEX06 datasets: mean test set error is shown (standard deviation is in brackets); D is the size of the feature space.

	INEX 2005		INEX 2006	
	D	Error	D	Error
C = 6	36894	6.25 (0.71)	49896	72.57 (2.86)
C = 8	54824	5.73 (0.76)	85536	73.57 (0.25)
C = 10	75570	5.23 (0.52)	130680	74.61 (0.43)

cation performance with respect to the Fisher kernel while consistently reducing the feature space size and, consequently, the computational cost of computing the kernel. While the simplest unigram representation does not seem capable of capturing enough discriminant information, already the bigram feature vector obtains results comparable to the Fisher kernel with as little as the 0.1% of the Fisher features. By enhancing the bigram vector with the unigram features (unibigram), Jaccard obtains a significant improvement with respect to Fisher, with a minor computational complexity. Such a behavior shows the high potential of introducing syntactical information in the adaptive kernel produced by the generative model, by means of the proposed multiset approach. On the same standard test set, the syntactic Subset Tree (SST) kernel [2] yields to an 11.21% classification error, which shows the efficacy of both generative kernels on the INEX 2005 dataset. On the INEX 2006 data, the proposed Jaccard kernel yields to a consistent reduction in the classification error with respect to Fisher, which pairs with a considerable reduction of the feature space size. However, the classification error remains above 70% for both models, while SST achieves 59.31%. This, seems due to a discrepancy between the cross-validation performance and the test-set. In the former, both models reach a validation error close to the 54%, which is not confirmed on the test samples. Notice that INEX 2006 seems particularly troublesome for the Fisher kernel. This often generates extremely dense undiscriminative Gram matrices, stretching the convergence time of SVM training (e.g. over 4 days for a single cross-validation trial) even for very small values of the cost parameter (due to the excessive number of support vectors selected). This has prevented obtaining a complete set of trials for the Fisher kernel, which motivates the large standard deviation for $C = 6$ on the INEX 2006 dataset, where only 2 trials out of 5 have lead to a successful completion of training, validation and test phases.

4 Conclusion

We have introduced a generative Jaccard kernel for the input conditional IO-BTHMM, with application to tree-structured data classification. Experimental results show that the proposed approach yields to a significant reduction in the classification error with respect to the Fisher Kernel, that pairs with a considerable contraction of the induced feature space. The proposed kernel defines a truly adaptive metric by virtue of the discriminative training of the IO-BHTMM model, which is used to learn probabilistic transductions of the input trees into their class representation. Nevertheless, the proposed approach is general under many aspects and can be applied to a large variety of problems and models. For instance, the multiset representation can be seamlessly employed with any probabilistic model using categorical latent variables, which easily extends the classes of data processable with the kernel. Further, the multiset representation allows to intuitively control the amount of syntactical information that is injected into the kernel, while controlling the tradeoff with computational complexity. Finally, by exploiting the general form of the Jaccard similarity in (1), it is possible to define more refined forms of structural matching, providing differentiated weights for hidden state agreements encountered in different positions of the structures, e.g. depending on the node level.

Acknowledgements

This work is partially supported by the FP7 RUBICON project (contract n. 269914). The work by A. Sperduti is supported by the Italian National project PRIN 2009 2009LNP494 005.

References

1. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. Advances in neural information processing systems (1999) 487–493
2. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: Proc. of the 40th Annual Meeting on Assoc. for Comput. Ling. (2002) 263–270
3. Bacciu, D., Micheli, A., Sperduti, A.: Input-output hidden markov models for trees. In Verleysen, M., ed.: Proc. of the 2012 Europ. Symp. on Artif. Neural Netw., Comput. Intell. and Machine Learn. (ESANN). (2012) 25–30
4. Jaccard, P.: The distribution of the flora in the alpine zone.1. New Phytologist **11**(2) (1912) 37–50
5. Nicotra, L., Micheli, A., Starita, A.: Fisher kernel for tree structured data. In: Proc. of the 2004 Int. Joint Conf. on Neural Netw. Volume 3. (2004) 1917 – 1922
6. Nicotra, L., Micheli, A.: Generative Kernels for Gene Function Prediction Through Probabilistic Tree Models of Evolution. Artificial Intelligence in Medicine (45) (2009) 125–134
7. Diligenti, M., Frasconi, P., Gori, M.: Hidden tree markov models for document image classification. IEEE Trans. Pattern Anal. Mach. Intell. **25**(4) (2003) 519–523
8. Jebara, T., Kondor, R., Howard, A.: Probability product kernels. The Journal of Machine Learning Research **5** (2004) 819–844
9. Denoyer, L., Gallinari, P.: Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents. SIGIR Forum **41**(1) (2007) 79–90
10. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.