Unsupervised learning of graph languages using kernels

Christophe Costa Florêncio
Department of Computer Science, K.U. Leuven,
Chris.CostaFlorencio@cs.kuleuven.be

May 10, 2007

1 Graph mining

In most domains, machine learning involves discovering the relationship between objects. Such relationships are naturally represented as *graphs*. Up until now, most work on learning graphs consisted of extending well-known approaches from the field of machine learning to deal with more complex structures.

For both technical and conceptual reasons, borrowing techniques from Grammar Induction (GI) seems an obvious step. In GI, learning a classification task is approached as learning a representation for a language, this representation can then be used to decide whether or not previously unseen data is in the target language. Such representations usually take the form of automata, grammars or rewriting systems, and these can be used to represent graph languages as well as string or tree languages. Some attempts in this direction have been made ([LS98, DHO02], for example), but much work still needs to be done.

A novel approach to GI, planar languages, makes use of linear equalities over a kernel feature space as representation. This way, inherently learnable language classes can easily be defined. Given the amount of existing literature on graph kernels, one would expect some expressive, well-understood kernels to be available that, with minor adjustments, can be used as the basis for the definition of rich learnable classes of graph languages.

2 Kernels and planar languages

The notion of *planar languages* was first defined in [CCFW06, CCFWS06]. These consist of sets of strings that correspond to data points that form a hyperplane in a feature space defined by a string kernel. Kernels are normally used in conjunction with an SVM or other classifier to perform supervised learning tasks. In the context of planar languages, the learning algorithm finds the lowest dimensional space spanned by the images of datapoints in feature space, this is a form of unsupervised learning.

It has been shown that all such language classes are learnable in the sense of being identifiable in the limit, with polynomial update time and given a very small dataset. Furthermore, the learner can work under constraints on its behaviour such as strong monotonicity.

2.1 Graphs

We define graphs as follows:

Definition 2.1 *A* graph $G = \langle V, E \rangle$ *consists of a set of* n *vertices* $V = \{v_1, \ldots, v_n\}$ *and a set of edges* $E \subseteq V \times V$.

We define the product graph, a useful concept for dealing with synchronous walks on two graphs:

Definition 2.2 *Let graphs $G = \langle V, E\rangle$ and $G' = \langle V', E'\rangle$ be given. A* (direct) product graph *$G_\times = \langle V_\times, E_\times\rangle$ is a graph with $|V| \cdot |V'|$ vertices, where each such vertex represents a pair of vertices from G and G', respectively. An edge $e \in E_\times$ connects two vertices in $G_\times$ just if the corresponding vertices in G and G' are adjacent in both these graphs.*

3 Graph kernels

In the context of planar languages, injectivity is a desirable property for kernels. In the case of string languages this is feasible, the gap-weighted string kernel for example has shown to be injective, for certain values of the decay value.

Unfortunately, in the case of graphs injectivity is not feasible. In [GFW03] it was shown that computing an injective graph kernel is at least as hard as deciding whether two graphs are isomorphic, a problem for which no polynomial time algorithm is known. This result has been strengthened in [RG03], where it was shown that even approximating such a kernel is hard.

3.1 The random walk graph kernel

In [BK07] a graph kernel is defined that is based on counting the number of matching random walks in graphs. Since this method does not limit itself to comparing local features, one might expect the bad performance that other graph kernels suffer from, typically $O(n^6)$ or worse. However, efficient and theoretically sound algorithms do exist. In [VBS06], a reduction to Sylvester equations yields algorithms with worst-case time-complexity of $O(n^3)$.

$$\kappa(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^\top W_\times^k p_\times \tag{1}$$

The function $\mu(k)$ is commonly defined as λ^k, with $\lambda > 0$. Given an appropriate value for λ, Equation 1 is well-defined.

3.2 The composite graph kernel

There is an asymmetry that is inherent in the concept of a product graph; absence of an edge between given vertices in both graphs G and G' is treated in the same way as absence of that edge in either G or G'. In some sense, potentially interesting information is ignored.

To counter this, [BK07] introduces a composite graph kernel based on the notion of *complement graph*. The complement graph $\overline{G} = \langle V, \overline{E}\rangle$ of a graph $G = \langle V, E\rangle$ has the same vertices as G, but just all the edges missing from G. The composite kernel is then defined as:

$$\kappa_{comp}(G, G') = \kappa(G, G') + \kappa(\overline{G}, \overline{G}') \tag{2}$$

The practical usefulness of this kernel has been established in [BK07], it outperforms the random walk graph kernel on real-world data (PPI and gene expression data).

4 Graph planar languages

The walk kernel can be taken as the basis of a planar language. Since this kernel is known not to be injective, it is necessary to investigate to what set of graphs a given point in feature space can map. We call any pair from the object space with this property a *congruent pair*.

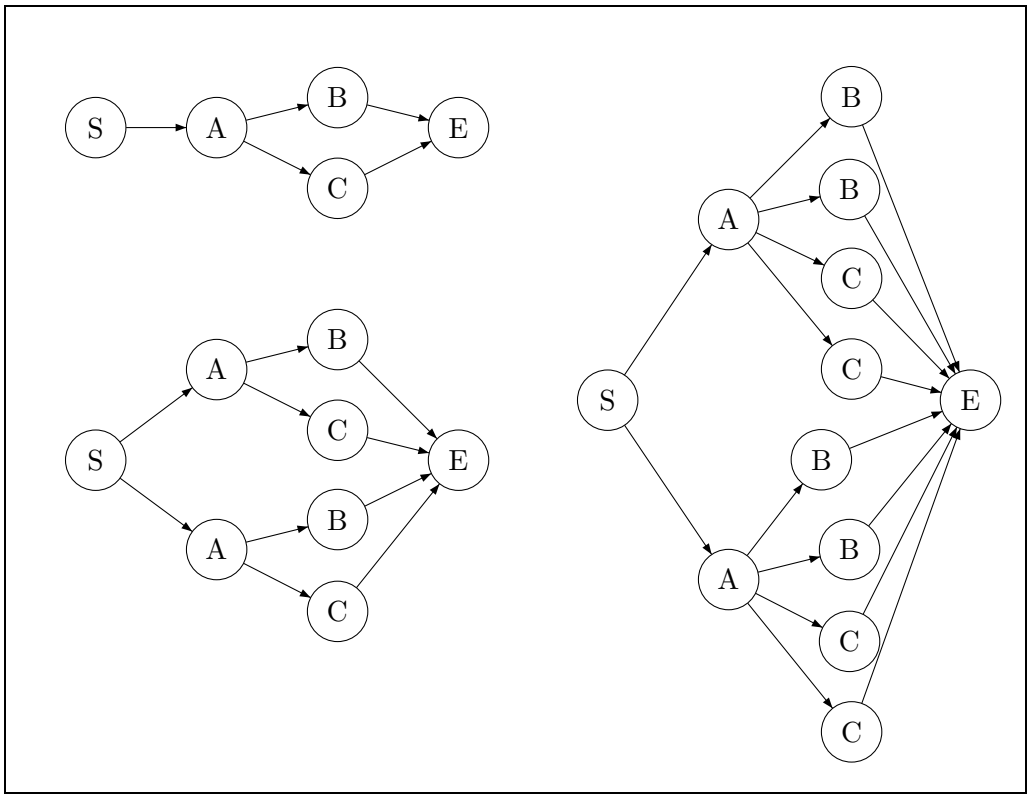

Figure 1: Graphs that map to the same point in walks feature space.

Consider the graphs in Figure 1, any two of these form a congruent pair for the walk kernel.[1] Let G be a directed, connected graph with some node n_1 with in-degree 0 and some node n_2 with out-degree 0. Let G_n be the graph consisting of G and n copies of the (acyclic) subgraph between n_1 and n_2, which are all connected to these nodes in the same way as in G. Then G and G_n map to the same point in the feature space spanned by the embedding underlying the walk kernel, for any n. It is easy to see that the exact same random walks exist for these graphs, and that they have the same likelihood. Figure 2 shows congruent pairs for the walk kernel where the graphs contain cycles. Note that the same examples hold for the composite kernel.

However, the composite kernel has greater expressive power; one can stipulate the non-existence of edges between nodes with certain labels by setting the values for all walks containing

[1]This example is a generalization of an example from [RG03] based on graphs made up of just 4 nodes.

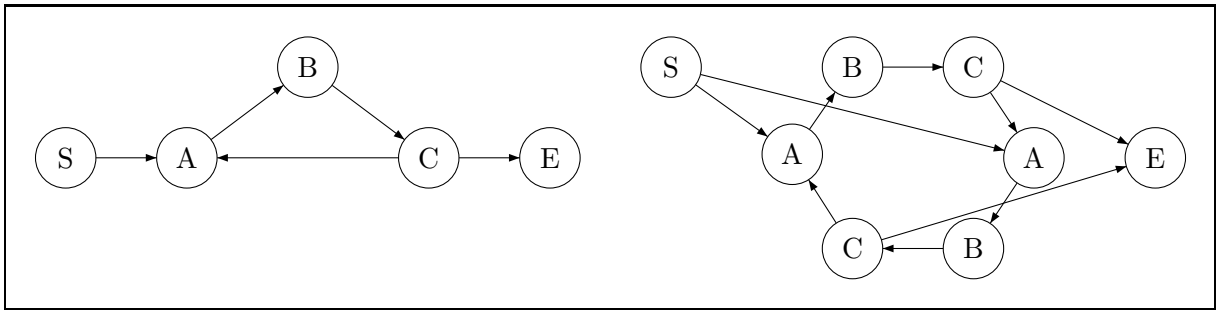

Figure 2: Cyclic graphs that map to the same point in walks feature space.

a step between two such nodes to zero. This is not possible for the walk kernel, thus if graph $G = \langle V, E \rangle$ is in a walk-planar language, then so is any graph $G' = \langle V, E \cup E' \rangle, E' \neq \emptyset$.

5 Conclusion

Planar languages offer a promising approach to learning rich classes of graph languages. It is expected that implementing efficient learning algorithms will pose little problems. Earlier experiments with a (string) planar language learner [CCFW06] indicate this approach is noise-sensitive, it would be interesting to see how a graph planar language learner performs on real-world data.

Although we only considered walk-based kernels here, there are others that are also good candidates for serving as a basis for graph planar languages, the tree-structured pattern kernel [RG03] for example.

References

[BK07] Karsten M. Borgwardt and Hans-Peter Kriegel. Graph kernels for disease outcome prediction from protein-protein interaction networks. In *Proc. of Pacific Symposium on Biocomputing (PSB 2007), Maui, USA*, 2007.

[CCFW06] Alexander Clark, Christophe Costa Florêncio, and Chris Watkins. Languages as hyperplanes: grammatical inference with string kernels. In *ECML, 17th European Conference on Machine Learning*. Springer-Verlag, 2006.

[CCFWS06] Alexander Clark, Christophe Costa Florêncio, Chris Watkins, and Mariette Serayet. Planar languages and learnability. In *International Colloquium on Grammatical Inference (ICGI)*, Tokyo, 2006.

[DHO02] Shailesh P. Doshi, Fang Huang, and Tim Oates. Inferring the structure of graph grammars from data. In *International Conference on Knowledge Based Computer Systems*, 2002.

[GFW03] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop*, pages 129–143. Springer-Verlag, August 2003.

[LS98] Damián López and José M. Sempere. Handwritten digit recognition through inferring graph grammars. A first approach. In *SSPR '98/SPR '98: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 483–491, London, UK, 1998. Springer-Verlag.

[RG03] Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In T. Washio and L. De Raedt, editors, *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.

[VBS06] S. V. N. Vishwanathan, K. Borgwardt, and N. N. Schraudolph. Fast computation of graph kernels. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*. MIT Press, Cambridge, MA, 2006.