

Completion of biological networks : the output kernel trees approach

Pierre Geurts^{1,2}, Nizar Touleimat^{1,3}, Marie Dutreix³, and Florence d'Alché-Buc¹

¹ IBISC FRE CNRS 2873 Epigenomics Project, GENOPOLE, Evry, France

² Dept. of EECS & CBIG/GIGA, University of Liège, Belgium

³ UMR 2027 CNRS-IC, Institut Curie, Orsay, France

Introduction

Elucidating biological networks appears nowadays as one of the most important challenge in systems biology. Due to the availability of various sources of data, machine learning has to play a major role regarding this issue, given its large spectrum of tools ranging from generative models to concept learning methods. In this work the focus is narrowed on the completion of biological interactions networks for which some of the interactions between variables (usually genes or proteins) are already known. Within this supervised framework, two related approaches have emerged: supervised methods that implicitly learn the mapping between some descriptors of a protein and its position as a node within the interactions graph [7, 6] and relational learning approaches concerned with the learning of the local concept of interaction between two proteins [1]. The former approaches provide a more global view of the problem, accounting for the specific nature of the output space using a graph-based kernel, whereas the latter approaches do not take into account the dependency between interactions. In this work, we have adopted the first supervised network inference framework introducing a new method based on a kernelisation of the outputs of regression tree based methods. This method recently introduced in [4] inherits several features of trees such as interpretability, robustness to irrelevant variables, and input scalability. Applied to the completion of a protein-protein interaction network, it provides relevant insights on input data regarding their potential relationship with the existence of interactions.

Supervised network completion

The problem of network completion has been introduced in [7] and subsequently considered in [6] as the supervised network inference problem.

Let $G = (V, E)$ be an undirected graph with vertices V and edges $E \subset V \times V$. $|V| = m$ is the number of nodes in the graph. We suppose that each vertex $v_i, i = 1..m$ can be described by some features in some input space \mathcal{X} , and we denote by $x(v_i) = x_i \in \mathcal{X}$ this information. Only the knowledge of a subgraph G_n of G is available during the training phase: without losing generality, we enumerate the nodes belonging to V_n as v_1, \dots, v_n where n is the number of nodes

in the subgraph denoted by $G_n = (V_n, E_n)$ with $V_n \subset V$ and $E_n = \{(v, v') \in E | v, v' \in V_n\}$. The goal of supervised graph completion is then to determine from the knowledge of G_n a function $e(x(v), x(v')) : V \times V \rightarrow \{0, 1\}$, ideally such that $e(x(v), x(v')) = 1 \Leftrightarrow (v, v') \in E$.

Our solution is based on the same kernel embedding of the graph as in [7]. We first define a kernel $k(v, v')$ such that adjacent vertices lead to high values of k and non-adjacent ones lead to smaller ones. The mapping ϕ of this kernel is thus such that $\phi(v)$ is close to $\phi(v')$ in \mathcal{H} as soon as v and v' are connected. Then, the problem of graph completion is solved by finding an approximation of the kernel from the known graph. A graph prediction is then obtained by connecting those vertices that correspond to a kernel prediction above some threshold.

Output kernel trees

Output kernel trees (OK3, [4]) are a kernelization of multiple output regression trees that allows this method to handle any structured output space over which a kernel may be defined and, by extension, to learn a kernel as a function of an input vector.

Multiple output regression trees are a straightforward extension of regression trees [2] that allows them to handle vectorial outputs. The idea of (multiple output) regression trees is to recursively split the learning sample with binary tests based on the input variables, trying at each split to reduce as much as possible the (empirical) variance of the output vector in the left and right subsamples of learning cases corresponding to that split.

For simplicity, let us assume that the mapping ϕ projects all vertices into a vector of \mathbb{R}^l , and that we have actually access to a learning sample $\{(x(v_1), \phi(v_1)), \dots, (x(v_n), \phi(v_n))\}$. Then, the idea is to apply multiple output regression trees on this learning sample to get an approximation $\hat{\phi}(x(v))$ of the output feature vector $\phi(v)$ corresponding to a new vertex v described by its input vector $x(v)$. The kernel value between two new vertices is then approximated by the dot-product of the predictions given by this model for the two vertices.

In this context, the score measure used to evaluate splits is written:

$$\text{Sc}(T, S) = \text{var}_{ls} - \frac{N_l}{N} \text{var}_{ls_l} - \frac{N_r}{N} \text{var}_{ls_r}, \text{ with } \text{var}_s = \frac{1}{N_s} \sum_{i=1}^{N_s} \|\phi(v_i) - \frac{1}{N_s} \sum_{i=1}^{N_s} \phi(v_i)\|^2 \quad (1)$$

where T is the split to evaluate, ls is the local learning sample of size N at the node to split, ls_l and ls_r are its left and right successors of size N_l and N_r respectively, and var_s denotes the variance of the output vector in a subset s of size N_s .

Once the tree is grown, each leaf L is labeled with a prediction $\hat{\phi}_L$ computed as $\hat{\phi}_L = \frac{1}{N_L} \sum_{i=1}^{N_L} \phi(v_i)$, where N_L is the number of learning cases that reach this leaf. From this tree, we want to make predictions about kernel values between two new vertices v and v' described by their input vectors $x(v)$ and $x(v')$. If

$x(v)$ (resp. $x(v')$) reaches leaf L_1 (resp. L_2) that contains vertices $\{v_1^1, \dots, v_{N_{L_1}}^1\}$ (resp. $\{v_1^2, \dots, v_{N_{L_2}}^2\}$), their inner product is approximated by:

$$\hat{k}(v, v') = \langle \hat{\phi}_{L_1}, \hat{\phi}_{L_2} \rangle = \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} \langle \phi(v_i^1), \phi(v_j^2) \rangle. \quad (2)$$

Both variance (1) and kernel predictions (2) are written only in terms of dot products and consequently in terms of kernel values only:

$$\text{var}_s = \frac{1}{n} \sum_{i=1}^n k(v_i, v_i) - \frac{1}{n^2} \sum_{i,j=1}^n k(v_i, v_j), \quad (3)$$

$$\hat{k}(v, v') = \frac{1}{N_{L_1} N_{L_2}} \sum_{i=1}^{N_{L_1}} \sum_{j=1}^{N_{L_2}} k(v_i^1, v_j^2). \quad (4)$$

Using the kernel trick, we can thus build a tree, from kernel values only, that implicitly try to find an approximation of the output feature vector as a function of the input vector and then can be used to make kernel predictions. We call this kernel formulation of the tree growing algorithm OK3, for output kernel trees. By construction, this method shares several features of standard tree-based methods, the most attractive ones being the interpretability of the model and the ability of the method to rank the features. OK3 can also benefit from tree-based ensemble methods. In our experiments, we built ensembles of OK3 with the extra-trees method proposed in [5].

Numerical experiments

We report here experiments concerning a Yeast protein-protein interaction (PPI) network. Further details and results on an enzyme network may be found in [3].

The PPI network was taken from [6]⁴. It consists of 2438 high confidence interactions that link 984 proteins. As the output kernel, we use the diffusion kernel $K = \exp(-\beta L)$, where L is the Laplacian of the graph and β a parameter that was fixed to 3.0. As input features, we combine the three sets of attributes considered in [6], i.e. gene expression data (157 variables), phylogenetic profiles (145 variables), localization data (23 variables), and yeast two hybrid (y2h) network. To obtain features from this latter kind of (pairwise) data, we compute the first 50 components obtained by applying kernel PCA on a diffusion kernel computed on the network.

We estimate the accuracy of our method by ten-fold cross-validation, predicting at each iteration all interactions that involve at least one protein from the test fold and varying the kernel threshold to obtain ROC curves. Table 1 gathers the average AUC values with different sets of variables. The last column reports

⁴ <http://www.cbrc.jp/kato/faem/faem.html>

Table 1. AUC results

Inputs	OK3	Kern.
expr	0.851	0.776
phy	0.693	0.767
loc	0.725	0.788
y2h	0.790	0.612
All	0.910	0.939

Table 2. Variable ranking

#	Att.	Imp.
1	loc - nucleolus	0.021
2	expr (Spell.) - elu 120	0.013
3	loc - cytoplasm	0.012
4	expr (Eisen) - sporulation ndt80 early	0.012
5	loc - nucleus	0.012
6	expr (Eisen) - sporulation 30m	0.011
7	expr (Eisen) - sporulation ndt80 middle	0.010
8	expr (Spell.) - alpha 14	0.010
9	expr (Spell.) - elu 150	0.010
10	loc - mitochondrion	0.009

the result obtained in [6] with the same protocol. The method in [6] exploits a kernel on the inputs with an algorithm based on expectation-maximization that automatically learns a weight for each data sources.

The results are quite good. The most important source of information is the expression data followed by the y2h network, localization data, and phylogenetic profiles. Combining all data sources allow to improve the AUC values with respect to each data source separately. When integrating all data sources, we get only slightly worse results than the method in [6]. We are doing a much better use of the expression data and the y2h network while this latter method is better in exploiting localization data and phylogenetic profiles.

One of the advantages of our tree-based approach is that it provides interpretable results to some extent. First, like standard trees, OK3 provides a partition of the learning sample into clusters, one for each tree leaf, where proteins are as much as possible connected between each other and each cluster is furthermore described by a rule based on the input variables (see [3] for an illustration). Second, working in the original input space, the method allows to rank the features according to their importance at predicting new edges. On this problem, the top ten variables of the ranking obtained without the y2h data (see Table 2) correspond exclusively to expression and localization features, confirming the ranking of the different data sources in Table 1.

Acknowledgments

Pierre Geurts is a postdoctoral researcher at the CNRS (France) and a scientific research worker at the FNRS (Belgium). Florence d’Alché-Buc thanks Genopole for funding her research activities in the context of an ATIGE funding.

References

1. A. Ben-Hur and W.S. Noble. Kernel methods for predicting protein-protein interactions. *Bioinformatics*, 21 Suppl.1:i38–i46, 2005.
2. L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone. *Classification and Regression Trees*. Wadsworth International, 1984.

3. P. Geurts, N. Touleimat, M. Dutreix, and d'Alché Buc. Inferring biological networks with output kernel trees. Technical report, University of Evry, IBISC, 2006.
4. P. Geurts, L. Wehenkel, and F. d'Alché Buc. Kernelizing the output of tree-based methods. To appear in *Proceedings of ICML*, 2006.
5. Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning Journal* (advance access: DOI 10.1007/s10994-006-6226-1), 2006.
6. T. Kato, K. Tsuda, and A. Kiyoshi. Selective integration of multiple biological data for supervised network inference. *Bioinformatics*, 21(10):2488–2495, 2005.
7. Y. Yamanishi and J.-P. Vert. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20:i363–i370, 2004.