Abstract

Summary: Measuring the similarity of graphs is a fundamental step in the analysis of graph-structured data, which is omnipresent in computational biology. Graph kernels have been proposed as a powerful and efficient approach to this problem of graph comparison. Here we provide graphkernels, the first R and Python graph kernel libraries including baseline kernels such as label histogram based kernels, classic graph kernels such as random walk based kernels, and the state-of-the-art Weisfeiler-Lehman graph kernel. The core of all graph kernels is implemented in C++ for efficiency. Using the kernel matrices computed by the package, we can easily perform tasks such as classification, regression and clustering on graph-structured samples.


Contact: mahito@nii.ac.jp or elisabetta.ghisu@bsse.ethz.ch

Supplementary information: Supplementary data are available online at Bioinformatics.

1 Introduction

Graph-structured data are steadily growing and extensively being analyzed in computational biology. For example, chemical compounds are modeled as graphs in drug discovery (Takigawa and Mamitsuka, 2013), and proteins are represented as graphs in protein function prediction (Dhilli and Nguif, 2015). Finding efficient solutions for measuring the similarity between a pair of graphs, known as the graph comparison problem, is a fundamental step in graph analysis, in order to perform classification or regression on graph data. There are two approaches to graph comparison: alignment-based methods (Faisal et al., 2015) that compare graphs via finding node mappings and alignment-free methods (Yaveroglu et al., 2015) that measure the similarity between graphs using features such as degree distributions or subgraph counts without identifying correspondences between nodes.

To date, among the alignment-free approaches, graph kernels have become a popular approach to quantify the similarity between graphs (Borgwardt and Kriegel, 2005; Costa and Grave, 2010; Gärtner et al., 2003; Kashima et al., 2003; Shervashidze et al., 2009, 2011; Sugiyama and Borgwardt, 2015; Vishwanathan et al., 2010), and are at the heart of many machine learning approaches in computational biology. A number of key applications of graph kernels exist such as biological function prediction from graph-based representations of chemical compounds. However, there is no convenient R or Python implementation that can simply and efficiently compute graph kernels, although R is a popular programming environment in Bioinformatics and Python in Machine Learning.

Here we present graphkernels, the first package in R and Python with efficient C++ implementations of various graph kernels including the following prominent kernel families: (i) simple kernels between vertex and/or edge label histograms, (ii) graphlet kernels, (iii) random walk kernels (popular baselines) and (iv) the Weisfeiler-Lehman graph kernel (state-of-the-art kernels). The packages can be easily used to perform graph classification and regression by machine learning algorithms (Fig. 1), such as support vector machines (SVMs) or the $k$-nearest neighbors algorithm.

2 Materials and methods

Each function implemented in the graphkernels packages receives a collection of graphs $G_1, G_2, \ldots, G_n$ and returns the kernel
(Gram) matrix \(K_{ij} \in \mathbb{R}^{m \times n}\) with the respective graph kernel, where each kernel value \(K_{ij}\) shows the similarity between graphs \(G_i\) and \(G_j\). The packages support the following 14 graph kernels:

- Linear kernels on label histograms: VertexHist, EdgeHist, VertexEdgeHist, VertexVertexEdgeHist.
- Gaussian RBF kernels on label histograms: VertexHistGauss, EdgeHistGauss, VertexEdgeHistGauss.
- Graphlet kernels: Graphlet, ConnectedGraphlet.
- The Weisfeiler-Lehman subtree kernel: WL.

All kernels are implemented in C++ and compiled through the packages Rcpp (Eddelbuettel, 2013) and RcppEigen (Bates and Eddelbuettel, 2013) in the R package. We use SWIG (Beazley, 1996) interfaces to wrap C++ code in Python. See the Supplementary Material for detailed mathematical definitions of these graph kernels.

In our packages, each graph is treated as an igraph object (Csardi and Nepusz, 2006) and a collection of graphs is kept as a list of igraph graphs. An example usage in R is shown in the following, where we use the dataset MUTAG (Debnath et al., 1991), a typical benchmark dataset that is also provided in our package.

```r
> library(graphkernels) # load the package
> data(mutag) # load a sample dataset,
# which is a list of (igraph) graphs
> K <- CalculateVertexHistKernel(mutag) # compute the kernel matrix
> K[1, 2]
[1] 282
# The kernel value b/w graphs 1 and 2
```

The entire kernel matrix can be easily computed by a single line of R code. Similar examples in Python can be found in the Supplementary Material.

### 3 Application

As a representative application, we demonstrate graph classification using the MUTAG dataset. In the dataset, there are 188 graphs, and the objective is to predict labels of graphs, indicating whether or not they are mutagenic. We used 10-fold cross validation for graph classification. We randomly divided the entire dataset into 10 folds. In each iteration 1 of the 10 folds was used for testing and the rest for training. We computed the kernel matrix of the training data using one of our functions implemented in graphkernels, and use these data to train an SVM using the kernlab package (Karatzoglou et al., 2004). We then predicted labels on the test data, and obtained the accuracy by comparison with the ground-truth labels. The detailed experimental methodology and the R code to reproduce these results are provided in the Supplementary Material.

Figure 2 shows the prediction accuracy for graph kernels in our package and the CPU running time needed to compute each kernel matrix. This example demonstrates that our package allows for an easy comparison of the effectiveness and the efficiency of various popular graph kernels and will serve as a baseline when designing new graph kernels for specialized applications in computational biology.

### Funding

MS was funded by JSPS KAKENHI Grant Numbers JP16K16115 and JP16H02870. MEG was funded by Horizon 2020 project CDS-QUAMRI, Grant No. 634541.

### References


