Graph kernels

Markus Heinonen

21.2.2008

◆□ ▶ ◆□ ▶ ◆三 ▶ ◆三 ▶ ● ○ ○ ○ ○

Graph kernels

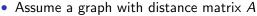
- Structured data is often represented as graphs, which comes in wide selection of types:
 - Phylogenetic trees
 - RNA structures
 - Gene regulation networks
 - Molecules
 - Natural language
 - XML
 - Protein representation as distance graphs
- Using graph-like data in kernel methods requires graph kernels.

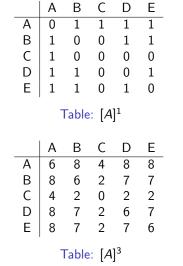
Graph kernels vs tree kernels

- Last week' tree kernel was defined on a feature vector of counts of subtrees or substrings. The definition was general and could be computed efficiently. Graph kernels are a generalization of tree kernels.
- Graph kernels presented in the literature are based on measure labeled walks. E.g.
 - Probability of walks (with equal labels), which is represented here (Kashima 2003). Infinite number of walks if graph is cyclic.
 - Counts of subgraphs (enumeration NP-hard) (Gartner 2003).
 - Counts of walks with equal labels containing gaps (Gartner 2003)
 - Counts of labeled walks with where first and last node are equal (Gartner 2003). This can be trivially computed using adjacency matrix exponential

 $[A']_{i,j} =$ counts of *l*-length walks from node *i* to *j*

Exponential kernels





• Kernel is based on vectors of walk counts.

Graph Notation

- A labeled directed graph G is a tuple $(\mathcal{X}, v, \mathcal{L}, e)$, where
 - \mathcal{X} is the set of nodes, function v maps nodes to labels
 - \mathcal{L} is the set of edges, function *e* maps edges to labels Graphs are simple.
- A walk **x** is a sequence of nodes $(x_1, x_2, ..., x_l)$ of length *l*, possibly infinite.
- A label sequence **h** is an alternating sequence of node and edge labels. Label sequence associated with walk **x** is

$$\mathbf{h}_{\mathbf{x}} = (v_{x_1}, e_{x_1, x_2}, v_{x_2}, \dots, v_{x_l}).$$

- ロ ト - 4 回 ト - 4 □ - 4

Walk probabilities

- Walks and label sequences have probabilities.
 - *p_s(x)* is the probability distribution of nodes to be the first node on a walk.
 - *p*_t(*x_i*|*x_{i-1}*) is the transition probability distribution from *x_{i-1}* to *x_i*.
 - *p_q(x)* is the probability distribution for the walk to end at node *x*.
- The probability of a walk x is

$$p(\mathbf{x}|G) = p_s(x_1) \prod_{i=2}^{l} p_t(x_i|x_{i-1}) p_q(x_l).$$

- ロ ト - 4 回 ト - 4 □ - 4

Label sequence probabilities

 The probability of a label sequence is the sum of probabilities of all walks emitting h

$$p(\mathbf{h}|G) = \sum_{\mathbf{x}} \mathbf{1}_{\mathbf{h}=\mathbf{h}_{\mathbf{x}}} \cdot p(\mathbf{x}|G)$$
$$= \sum_{\mathbf{x}} \mathbf{1}_{\mathbf{h}=\mathbf{h}_{\mathbf{x}}} \cdot \left(p_s(x_1) \prod_{i=2}^{l} p_t(x_i|x_{i-1}) p_q(x_l) \right)$$

 On a graph where each label is distinct, there's only one walk generating each label sequence. Doesn't apply in general case (e.g. molecular graphs).

Label sequence kernels

 The kernel for label sequences is a pair-wise product of label kernels

$$k_z(\mathbf{h},\mathbf{h}') = k_v(h_1,h_1') \prod_{i=2}^{l} k_e(h_{2i-2},h_{2i-2}') k_v(h_{2i-1},h_{2i-1}'),$$

where k_v is a kernel for nodes and k_e is a kernel for edges. Kernel k_z is zero if sequences have differing lengths.

- Valid kernels k_v and k_e can be chosen appropriately. For example
 - Identity kernel: $k_v = 1_{v=v'}$.
 - Gaussian kernel if the labels are real valued.

General graph kernel

The graph kernel is defined as the expectation of kernel k_z over all possible h and h'

$$k(G, G') = \sum_{\mathbf{h}} \sum_{\mathbf{h}'} p(\mathbf{h}|G) p(\mathbf{h}'|G') \cdot k_z(\mathbf{h}, \mathbf{h}').$$

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへぐ

Kernel's feature space

- The kernel's features are label sequences.
- In directed acyclic graphs feature space is limited. This case is computed using recursive definition of the problem with dynamic programming.
- In general case (cyclic graphs) feature space is possibly infinite because of loops.
- The computation of cyclic graph kernel can still be done with linear system theory and convergence properties of the kernel.

Reformulation

- For further equations, let's define pair-wise partial kernel values:
 - $s(x_1, x'_1) = p_s(x_1)p'_s(x'_1)k_v(v_{x_1}, v_{x'_1})$ • $t(x_1, x'_1, x_{i-1}, x'_{i-1}) = p_t(x_1|x_{i-1})p'_t(x'_i|x'_{i-1})k_v(v_{x_i}, v'_{x'_i})k_e(e_{x_{i-1}x_i}, e_{x'_{i-1}x'_i})$ • $q(x_i, x'_i) = p_q(x_i)p'_q(x'_i)$
- Term *s* is the partial kernel value of the first node in the label sequence over all possible first nodes.
- Term *t* is the partial kernel value for transition to next node.

・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・

• Term *q* is the partial kernel value for the last node in sequence.

Computation of the graph kernel in acyclic case

 In the case of directed acyclic graphs the nodes can be topologically ordered such that there is no path from node j to i if i < j. Kernel can be redefined as

$$k(G, G') = \sum_{x_1, x_1'} s(x_1, x_1') r(x_1, x_1'),$$

where r is a recursive function

$$r(x_1, x_1') = q(x_1, x_1') + \sum_{j > x_1, j' > x_1'} t(j, j', x_1, x_1') r(j, j').$$

Since nodes are ordered, the sum iterates over smaller and smaller set. Dynamic programming handles this problem in time $O(|\mathcal{X}| \cdot |\mathcal{X}'|)$.

Computation in cyclic case

• Let's first define r_l as the partial kernel value for *l*-length label sequences with first node x_1 and x'_1 . Only thing missing from the value is first node kernel values.

$$r_{l}(x_{1}, x_{1}') = \left(\sum_{x_{2}, x_{2}'} t(x_{2}, x_{2}', x_{1}, x_{1}') \left(\cdots \left(\sum_{x_{l}, x_{l}'} t(x_{l}, x_{l}', x_{l-1}, x_{l-1}') q(x_{l}, x_{l}') \right) \right) \right).$$

Now the kernel can be formulated as

$$k(G, G') = \sum_{x_1, x_1'} s(x_1, x_1') \underbrace{\sum_{l=1}^{\inf} r_l(x_1, x_1')}_{R_{\inf}(x_1, x_1')}.$$

To compute the $R_{inf}(x_1, x'_1)$ we have following system of linear equations

$$R_{\inf}(x_1, x_1') = \underbrace{r_1(x_1, x_1')}_{q(x_1, x_1')} + \sum_{i,j} t(i, j, x_1, x_1') R_{\inf}(i, j).$$

R_L is perceiveed as discrete time linear system. *R_L* is converging, and thus we solve the linear equations.

Matrix computation

The linear equation system can be computed using matrix notation with

$$k(G,G')=(I-T)^{-1}\mathbf{r}_1\mathbf{s}.$$

Here

- $\mathbf{s} = (\cdots, s(i, j), \cdots)^T$ • $\mathbf{r}_1 = (\cdots, r_1(i, j), \cdots)^T$
- T is the transition probability matrix
- Computing the kernel requires solving linear equation with $|\mathcal{X}|^2 \times |\mathcal{X}|^2$ coefficients.

・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・

Variants

 Kernel can be modified with weight decay, where the probabilities of label sequences decay with the length of sequence. The decay parameter λ_k is appended to transition probabilities p_t(x_i|x_{i-1}):

 $\lambda_k p_t(x_i|x_{i-1}),$

- ロ ト - 4 回 ト - 4 □ - 4

where k is the length of the walk so far.

Experiments

- Graph kernel was used with SVM to classify molecule toxicality. Method was compared with Pattern Discovery (PD) algorithm
- PD identifies all label sequences which appear in at least *m* graphs. With relatively high values of *m* PD finds small set of significant features. PD is complex and computationally prohibitive for lower values of *m*.
- Both methods were of comparable classification accuracy.
- Graph kernel parameters were set to uniform probabilities for p_s and p_t and a constant for p_q .

Summary

- A kernel for general graphs based on labeled walks (label sequences).
- Lengths of the walks can be modified with p_t and p_q parameters. Likelihood for long walks can be reduced further with decay term λ_k.
- Kernel extracts structural information of the graph. Difference with e.g. subgraph-based kernel?

References

- Kashima, H., Tsuda, K., Inokuchi, A.: Kernels for Graphs. *Kernel methods in computational biology* (2004), pp. 155-170.
- Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning* (2003), pp 321-328.
- Gartner, T.: Exponential and geometric kernels for graphs. In NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data (2000).
- Gartner, T., Flach, P., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the Seventh Annual Workshop on Kernel Machines* (2003).