Inferring vertex properties from topology in large networks

Janne Aukia[*], *Samuel Kaski*[†], *and Janne Sinkkonen*[*]
[*]Xtract Ltd, Helsinki, Finland
[†]Laboratory of Computer and Information Science, Helsinki University of Technology, Finland

Data collections representable as networks appear now frequently in many fields, and inferring properties of the vertices from the relations has become a common data mining problem. We introduce a model for the case where no *a priori* defined characteristics for the vertices are available, and the task is to infer latent vertex or edge properties from the topology. Depending on the hyperparameters, the properties are either latent blocks ("communities" in social networks) or more graded, hidden-variable like structures. As a demonstration, musical tastes of people are induced from the friendship network of the online music service Last.fm (`www.last.fm`), with over 10^5 vertices and edges.

The model generates *edges*, and is fitted with collapsed Gibbs sampling. Mixture components have a Dirichlet process prior. The model structure allows a sparse implementation, with which analysis of millions of vertices with models including thousands of latent components come feasible.

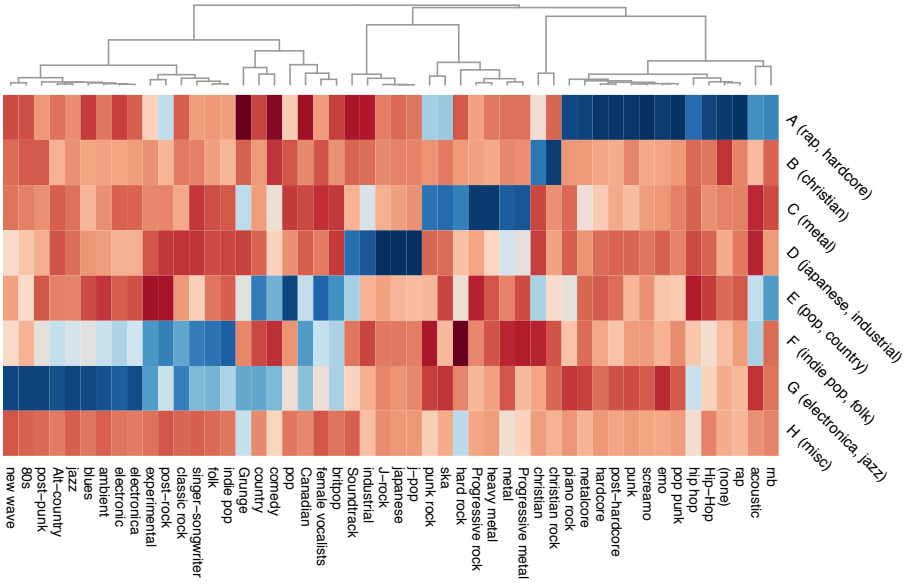
The generative process out of which the network is supposed to arise is the following; it is parameterized by (α, β): (1.1) Initialize by generating a multinomial distribution θ_z over an infinite number of *latent components* z, generated by the Dirichlet process $\mathrm{DP}(Z|\alpha)$; (1.2) to each z, assign a multinomial distribution over the M vertices i by sampling the multinomial parameters m_{zi} from the Dirichlet distribution $\mathrm{Dir}(\beta)$—the prior is constant over the vertices. (To clarify, we have $\sum_i m_{zi} = 1$ for each z, and $\sum_z \theta_z = 1$.); (2) Then, repeat for each edge l: (2.1) draw a latent component z from the multinomial θ_z; (2.2) generate two vertices, i and j, independently of each other, with probabilities m_z; set up a non-directed edge between i and j.

Note that within components edges are generated independently of each other and "randomly"; the non-random structure of the network emerges from the tendency of components to prefer certain vertices (that is, m_z). There is no explicit hierarchy level for vertices, but because vertices typically have several edges, they are implicitly treated as mixtures over the latent components. Finally, the model, in its current form, is parameterized to generate self-links and multi-edges although they are not present in typical data sets. This allows sparse implementations that would be impossible for an equivalent Bernoulli model.

Although the number of potentially generated components is infinite, the Dirichlet process gives a very uneven distribution over them. With a suitably small value of α, we observe much fewer components there is edges, and the model is useful. On the other hand, β describes the unevenness of the degree distribution of the vertices *within components*: a high β tends to distribute edges evenly over the vertices, and therefore give components spanning over all vertices, while a small β prefers mutually exclusive, community-like components.

In the collapsed Gibbs sampling algorithm, the unknown model parameters m_{zi} and θ_z are marginalized away, and only the latent classes of the edges, z_l, are sampled one at a time.

Given the simplicity of the sampling scheme, it is easy to make it highly effi-



cient. For example, the degree of a vertex poses an upper limit for its component heterogeneity, so that in most real-life networks only few of the counts k_{zi} are simultaneously non-zero, allowing sparse implementation and a high number of components. (We have experimented with 10^6 vertices and 10^4 components.) Edges can be sampled in parallel by locking vertices in the count table k_{zi}. For M vertices, L edges, C components, and I iteration rounds, memory consumption scales as $O(MC + L + C)$, but with hash tables this can be reduced to $O(Md + L + C)$ where d is the average degree. Because $d = L/M$, memory consumption scales as $O(L + C)$. After optimizing the sampling with trees, running time can be lowered down to $O(I\,L\,d\,\log C)$. That is, running time scales linearly in the number of edges and logarithmically in the number of components—excluding the required number of iterations I.

In the figure, the component structure reflecting musical tastes resulted from a run with 147,610 (anonymous) Last.fm users claiming to be from US, and their 352,987 self-announced mutual friendships. The parameters of the model (α=0.2, β=0.2) were chosen to prefer a small number of diffuse components. As a result (10,000 iterations in just under four hours), each user gets a probabilistic profile over eight latent components. The components were afterwards correlated with user's listening habits; in the figure songs are aggregated by tags given to them. Users have mixed tastes, while tags are intuitively grouped into components. The tastes emerge from the friendships, which makes the approach usable in customer relationship management and personalization tools.