Superawesome Graph Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Nulla pulvinar ante rutrum efficitur pellentesque. Donec augue tortor, dapibus vitae quam at, ultricies bibendum mauris. Etiam est nisi, ultricies vitae est euismod, rutrum eleifend ligula. Donec dictum orci ullamcorper ipsum vulputate gravida. Phasellus dignissim commodo feugiat. Proin bibendum interdum malesuada. Phasellus nibh dolor, pulvinar vel hendrerit vitae, egestas eu mi. Aenean ornare mauris purus, porta euismod turpis tristique eget. Maecenas magna velit, tempus aliquet nunc quis, vulputate faucibus dui. Cras egestas viverra libero faucibus vehicula. Donec molestie lectus in mattis convallis.

1 Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tempus nunc tempus, pretium leo sed, blandit ante. Nulla sit amet consectetur nunc. Curabitur dictum, risus in eleifend tincidunt, ipsum libero dapibus magna, eu lacinia purus ligula ut lorem. Sed vel eleifend massa, eget tincidunt erat. Nulla vel semper ipsum, quis tincidunt dolor. Duis convallis diam at auctor laoreet. Maecenas at lorem egestas, mollis felis sit amet, sagittis est. Sed eget lorem ut sapien porttitor malesuada. Fusce vehicula, nunc sit amet auctor ultrices, odio ligula gravida enim, sed bibendum risus quam sit amet justo. Donec eget mauris at tortor iaculis finibus. Suspendisse sollicitudin tellus quam, eget aliquam ipsum mollis vitae. Morbi et hendrerit eros, ut consequat justo. Ut blandit lobortis sem. Donec sem felis, scelerisque tincidunt nibh sit amet, eleifend consectetur magna. Nunc ultricies gravida commodo. Morbi vestibulum tellus nec turpis imperdiet tempor.

2 Method

Previously Costa introduced a method to learn how to construct novel graphs. Graphs would be vectorized via a *decomposition kernel* to train a machine learning model (e.g. an SVM). Also fragments of the Graphs would be collected in a *grammar* (resembling a string grammar) to alter the set of Graphs incrementally. Changes to a graph are evaluated with the model. We present a method to increase the flexibility of the graph grammar.

Modification to the grammar. We work with different CIPs that consider a contracted version of the original graphs. We obtain a contracted graph G' by contracting edges in G. After a contraction, the set of contracted vertices of the created vertex is accessible with the $contracted$ function. We extract $C^v_R(G')$ and $I^v_{R,T}(G')$ as usual, but from G'. The core graph $C^v_R(G', G)$ is induced by the nodes $\bigcup_{u \in C^v_R(G')} contracted(u)$. The new interface graph $I^v_{R,B}(G', G)$ is then obtained by the nodes $\{w | d(w, v) \leq B \wedge v \in C^v_R(G', G) \wedge w \in G \wedge w \notin C^v_R(G', G)\}$. B is the thickness of the base graph. At this point we can construct a CIP from $C^v_R(G', G)$ and $I^v_{R,B}(G', G)$. To find a congruent CIP, we previously only compared the hashed $I^v_{R,T}(G)$ graphs. By hashing the hashes of $I^v_{R,T}(G')$ and $I^v_{R,B}(G, G')$ we increase the specificity of this comparison. The vertices in $I^v_{R,B}(G, G')$ might have

Submitted to 30th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

been relabeled to represent a concept of its $contracted$ set. In our test, we will contract according to the secondary RNA structure and label the resulting vertex accordingly e.g. 'Hairpin loop'. This way we encode far reaching and abstract in our CIP interface matching.

An improvement to the notion of congruency. For CIPs to be congruent, isomorphism is required. To cover some corner cases we expand this requirement to incorporate the distance to nodes in the core graph when determining if graphs are isomorphic. $\forall u \in I^v_{R,T}(G) : \min_{z \in C^v_R(G)} d(u,z) = \min_{z' \in C^{v'}_R(G')} d(\phi(u), z')$ i.e. the distance to the closest core node is equal for every u and $\phi(u)$.

Extending what we can contract. Looking at edge contraction is only one way to obtain a contraction graph. One might also contract nodes that are not connected by an edge. **??? what did i do exactly for the t-rna ??? how did i do the directedness stuff??**

3 Evaluation

RNA and their structure is subject to change in the course of evolution. RNAs are grouped into functional families whose classification is a problem of biology. *Infernal* can classify and create members of these families using covariance models. Infernal gives us a domain specific way to evaluate our Method for the generation of new graphs.

We set up two experiments to evaluate the success of our method. First we compare our generated sequences to the Infernal model. The Infernal model is human optimized and should be highly relyable. In a second experiment we observe the influence of adding GraphLearn generated graphs to a learning curve.

We evaluated our results on secondary RNA structure graphs, where each nucleotide is a vertex. The contracted graphs are obtained by contracting vertices that belong to the same structural element. E.g. adjacend stem vertices are contracted into a single vertex etc. Secondary structures are obtained via **rna folding thing**. For RNA sequences, the direction in which a sequence is read is important. Working with directed graphs in the grammar will enable us to extract the sequence along the backbone easily as well as adapt the model to the directed biological reality.

We chose our RNA sequences from the seed sequences of each family that infernal is using. Sequences were chosem from RNA families with a) many members, because some families only contain 10 instances which is very few for a classification task. b) interesting structure, many sequences fold into structures that exhibit a large number of unpaired bases which would result in a very simple contracted graph c) similar length, because classification should no be completely obvious.

4 Discussion