

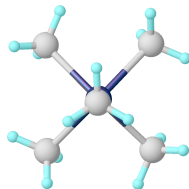
Learning on Graphs for Biological Problems

Carlos G. Oliver

ML Reading Group

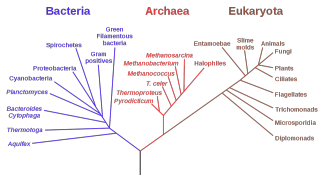
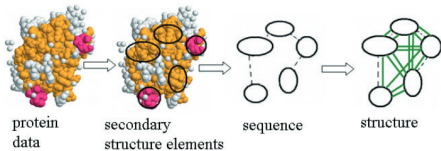
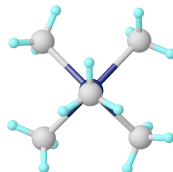
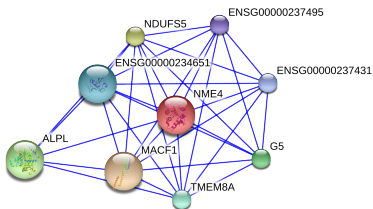
July 4, 2019

- 1 Why Are Graphs Interesting?
- 2 Pre-neural net models
- 3 Graph Convolutional Networks
- 4 Drug Design with GCNs



Graphs in [Bio/chem]informatics

- 1 Many 'biological' objects are naturally structured.



- Representation Problem
 - Standard predictors require fixed-size vectors as input (i.e. feature vectors)
 - Graphs (or subgraphs) are variable in size
 - We often lack a notion of explicit 'features' that captures structure
- Solutions
 - Implicit → Graph Kernels
 - Explicit → Dissimilarity Embedding
 - Learned → Graph Convolutional Networks

- Kernel-based predictors work without explicit feature maps.
- Instead of explicitly defining features, we define a similarity (kernel) function over graphs [Vishwanathan et al., 2010]

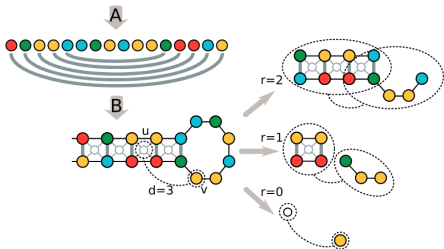
$$k(G, G') = \langle \phi(G), \phi(G') \rangle \quad (1)$$

- All requirements of kernels apply

Neighbourhood Overlap

- Compare neighbourhoods between nodes [Heyne et al., 2012]
- Used to identify clusters of sub-structures in RNA 2D structures.
- Decomposition Kernels: graph kernel k function of kernel κ on sub-graphs.
- κ checks isomorphism between pairs of subgraphs in G and G' , k aggregates.

ACUUGGCUGUUCAAGU
 (((((.))))))

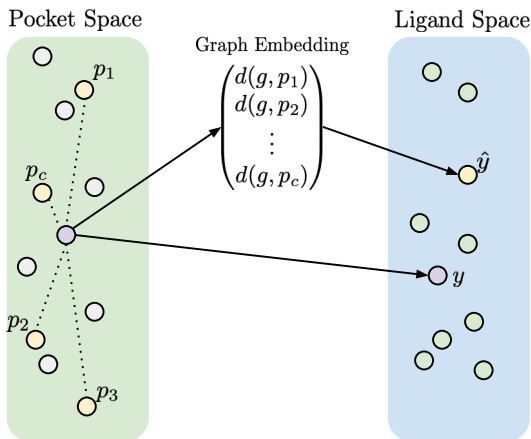


$$\kappa_{r,d}(G, G') = \sum \mathbb{1}(A \cong A') \mathbb{1}(B \cong B')$$

$$K(G, G') = \sum_r \sum_d \kappa_{r,d}(G, G')$$

Dissimilarity Embedding

- A graph is represented by its distance to a fixed set of graphs [Riesen and Bunke, 2010]
- Given a graph distance function d and a fixed set of data points P , we get a vector representation $\phi(g) \in \mathbb{R}^{|P|}$ of g as $\phi(g)_i = d(g, P_i)$



Graph Convolutional Networks (GCN)

Goal: a vector representation of nodes and graphs.

- **Idea:** let the input graph(s) define the neural network architecture.
- The representation of a node depends on the representations of its neighbors.
- 'Convolutonal' because we apply the same transformation to all neighbourhoods followed by pooling.

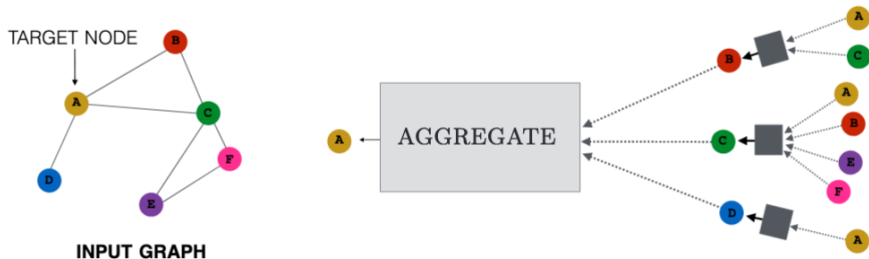


Figure: Schematic of neighbourhood aggregation. [Hamilton et al., 2017b]

Interesting biological applications

- Convolutional networks on graphs for learning molecular fingerprints [Duvenaud et al., 2015]
- Towards gene expression convolutions using gene interaction graphs. [Dutil et al., 2018]
- Protein interface prediction using graph convolutional networks [Fout et al., 2017]

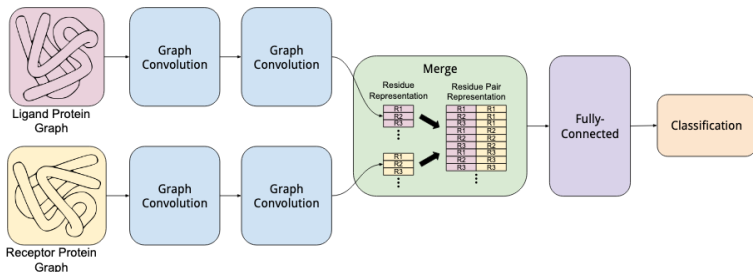


Figure: GCN model from [Fout et al., 2017] for predicting interface residues in PPIs.

Message Passing Framework

- Gilmer et.al. [Gilmer et al., 2017] define a general framework for describing the many proposed architectures.
- Node information \rightarrow message
- Transform messages from neighbors to compute hidden representation.
- Use hidden representations to make predictions.
- Two phases: (1) Message Passing, (2) Readout

Message Passing (1)

- At $t = 0$, $m_v^t = x_v$ for all $v \in G$
- Sum over function M_t applied to each neighbor and self.
- Compute message m_{t+1}

$$m_v^{t+1} = \sum_{w \in \mathcal{N}(v)} M_t(h_v^t, h_w^t, e_{vw}) \quad (4)$$

Message Passing (2)

- Once we have our aggregated messages, we can apply U_t to get hidden states.

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \quad (5)$$

- U_t takes message and current hidden state and transforms to obtain hidden state at next layer.
- At ever t messages from neighbours at 1 more 'hop' are incorporated.

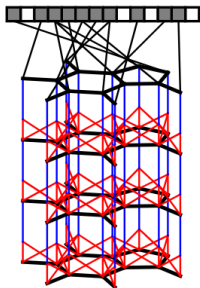
- Finally, we get a graph representation.

$$\hat{y} = R(\{h_v^T | v \in G\}) \quad (6)$$

- R operates on all nodes to produce final representation \hat{y}
- e.g. R is the average over all h_v
- Since M_t , U_t , R are differentiable losses can be backpropagated.
- Can optionally train directly on h_v for node-level tasks.

Example: Molecular Fingerprints [Duvenaud et al., 2015]

- Fingerprints are fixed-size vector representations of chemicals.
- GCNs 'invented' to produce smooth/continuous distribution of fingerprints.



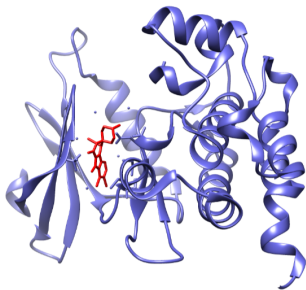
- $M_t = \text{CONCAT}(h_v, h_w, e_{vw})$
- $U_t = \sigma(H_t m_v^{t+1})$
- $R = \text{FullyConnected}(\sum_{v,t} \text{SOFTMAX}(W_t h_v^t))$

Algorithm 2 Neural graph fingerprints

- 1: **Input:** molecule, radius R , **hidden weights** $H_1^1 \dots H_R^5$, **output weights** $W_1 \dots W_R$
 - 2: **Initialize:** fingerprint vector $\mathbf{f} \leftarrow \mathbf{0}_S$
 - 3: **for** each atom a in molecule
 - 4: $\mathbf{r}_a \leftarrow g(a)$ ▷ lookup atom features
 - 5: **for** $L = 1$ to R ▷ for each layer
 - 6: **for** each atom a in molecule
 - 7: $\mathbf{r}_1 \dots \mathbf{r}_N = \text{neighbors}(a)$
 - 8: $\mathbf{v} \leftarrow \mathbf{r}_a + \sum_{i=1}^N \mathbf{r}_i$ ▷ sum
 - 9: $\mathbf{r}_a \leftarrow \sigma(\mathbf{v} H_L^N)$ ▷ smooth function
 - 10: $\mathbf{i} \leftarrow \text{softmax}(\mathbf{r}_a W_L)$ ▷ sparsify
 - 11: $\mathbf{f} \leftarrow \mathbf{f} + \mathbf{i}$ ▷ add to fingerprint
 - 12: **Return:** **real-valued** vector \mathbf{f}
-

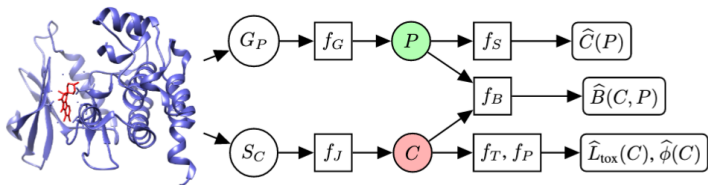
Latent Molecular Optimization for Targeted Therapeutic Design [Aumentado-Armstrong, 2018]

- **Problem:** given a **target** find the drug (compound) most likely to bind **and** have desired properties.
- **Brute force:** Try all compounds for a target: $\rightarrow 10^{24}$ possible compounds [Ertl, 2003].
- **Idea:** Use GCNs to encode target and ligand structure and predict compounds with desirable properties.
- Related approach [Mallet et al., 2019]



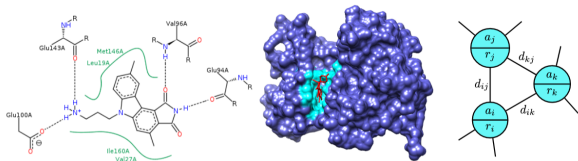
Pipeline

- Training input: protein(G_P)/ligand (S_C) complex
- P is a vector embedding of a protein graph using f_G which is a GCN
- C is an embedding of the compound using string encoder f_J



- Given P we predict a compound $\hat{C}(P)$
- Given P **and** C we predict $\hat{B}(C, P)$ binding strength
- Given C we predict $\hat{L}(C)$ toxicity, and drug-likeness $\hat{\phi}(C)$

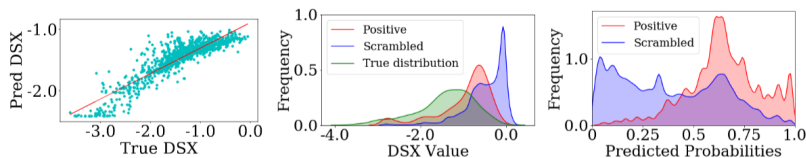
Binding Site Representation



- Nodes are atoms, edges are inter-atomic interactions weighted by distance
- Compute hidden states for each node in matrix.
- $h_v^{(0)}$ is vector of node features.
- Message function: $M_t(h_v^t, h_w^t) = (\text{deg}(v)\text{deg}(w))^{-\frac{1}{2}} A_{vw}$
- Update function: $U_t(h_v^t, m_v^{t+1}) = \text{ReLU}(W^t m_v^{t+1})$
- Readout: $R = \text{FullyConnected}(\sum_{v,t} \text{SOFTMAX}(\widetilde{W}_t h_v^t))$
- The result is a vector representation of the protein

Affinity Prediction

- Use DrugScoreX (DSX) scoring function to 'label' Protein-Ligand Complexes (PLCs) with a binding strength.
- Predict the strength of binding and the probability of binding from C



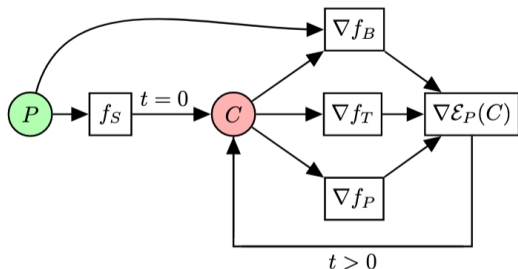
- Not shown: they are also able to predict toxicity and synthetic accessibility.

Latent Space Optimization

- Once the model is trained, we can explore the latent space to improve the predicted compound directly $C = f_S(P)$.
- For a fixed target protein P , and variable compound C we define an energy function \mathcal{E}_P .

$$\mathcal{E}_P(C) = E_B(C, P) + E_P(C) \quad (7)$$

- The energy function is a tradeoff between binding strength $E_B(C, P)$ and desired chemical properties $E_P(C)$



Algorithm 1 Molecular Optimization

```
1: procedure LATENTOPT( $P, T, \eta$ )
2:    $C_0 = f_S(P)$ 
3:   for  $t = 1$  to  $T$  do
4:      $v \leftarrow \nabla \mathcal{E}_P(C_{t-1})$ 
5:      $C_t \leftarrow \text{ADAM}(C_{t-1}, \eta, v)$ 
6:   end for
7:   return  $C_T$ 
8: end procedure
```

Optimization Results

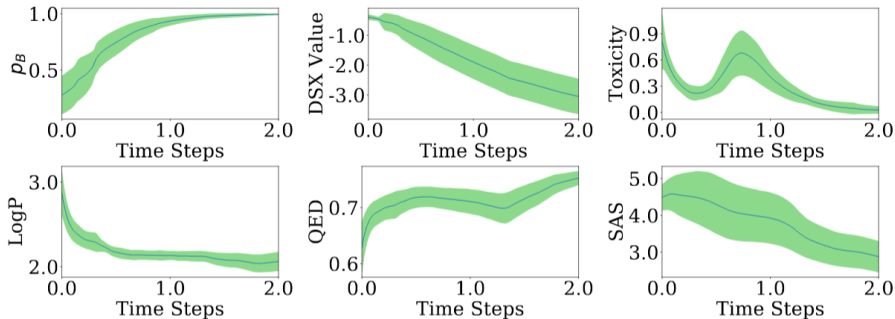


Figure: Improvement of compound after optimization process.

Optimization Results

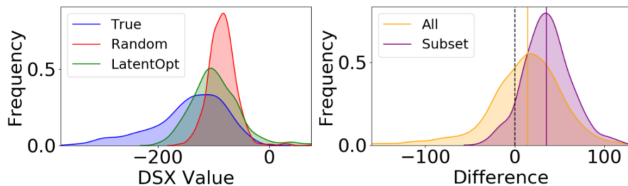
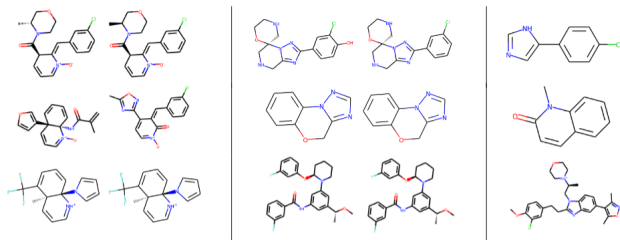





Figure: Left: predicted compound $C = f_s(P)$, middle: optimized compound, right: true ligand. Bottom: DSX scores from docking optimized vs random compounds.




- ① Graphs are very useful in biology
- ② Kernel methods were first attempt at learning on graphs but require manual construction which can lead to bias.
- ③ Continuous representations are improving and they allow for very efficient explorations of structured spaces.




- GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks [Ying et al., 2019]
- How powerful are Graph Neural Networks [Xu et al., 2018]
- Inductive representation learning on large graphs [Hamilton et al., 2017a]
- DEFactor: Differentiable Edge Factorization-based Probabilistic Graph Generation. [Assouel et al., 2018]

References I

-  Assouel, R., Ahmed, M., Segler, M. H., Saffari, A., and Bengio, Y. (2018). Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*.
-  Aumentado-Armstrong, T. (2018). Latent molecular optimization for targeted therapeutic design. *arXiv preprint arXiv:1809.02032*.
-  Dutil, F., Cohen, J. P., Weiss, M., Derevyanko, G., and Bengio, Y. (2018). Towards gene expression convolutions using gene interaction graphs. *arXiv preprint arXiv:1806.06975*.

References II

-  Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.
-  Ertl, P. (2003). Cheminformatics analysis of organic substituents: identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups. *Journal of chemical information and computer sciences*, 43(2):374–380.
-  Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A. (2017). Protein interface prediction using graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 6530–6539.

-  Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017).
Neural message passing for quantum chemistry.
In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org.
-  Hamilton, W., Ying, Z., and Leskovec, J. (2017a).
Inductive representation learning on large graphs.
In Advances in Neural Information Processing Systems, pages 1024–1034.
-  Hamilton, W. L., Ying, R., and Leskovec, J. (2017b).
Representation learning on graphs: Methods and applications.
arXiv preprint arXiv:1709.05584.






Heyne, S., Costa, F., Rose, D., and Backofen, R. (2012).
Graphclust: alignment-free structural clustering of local rna secondary structures.
Bioinformatics, 28(12):i224–i232.



Mallet, V., Oliver, C. G., Moitessier, N., and Waldispuhl, J. (2019).
Leveraging binding-site structure for drug discovery with point-cloud methods.
arXiv preprint arXiv:1905.12033.



Riesen, K. and Bunke, H. (2010).
Graph classification and clustering based on vector space embedding.
World Scientific.

-  Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010).
Graph kernels.
Journal of Machine Learning Research, 11(Apr):1201–1242.
-  Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018).
How powerful are graph neural networks?
arXiv preprint arXiv:1810.00826.
-  Ying, R., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019).
Gnn explainer: A tool for post-hoc explanation of graph neural networks.
arXiv preprint arXiv:1903.03894.