Graph Memory Networks for Molecular Activity Prediction

Trang Pham, Truyen Tran, Svetha Venkatesh
Centre for Pattern Recognition and Data Analytics
Deakin University, Geelong, Australia
{*phtra, truyen.tran, svetha.venkatesh*}*@deakin.edu.au*

Abstract

Molecular activity prediction is critical in drug design. Machine learning techniques such as kernel methods and random forests have been successful for this task. These models require fixed-size feature vectors as input while the molecules are variable in size and structure. As a result, fixed-sized fingerprint representation is poor in handling substructures for large molecules. Here we approach the problem through deep neural networks as they are flexible in modeling structured data such as grids, sequences and graphs. We propose Graph Memory Network (GraphMem), a memory-augmented neural network to model the graph structure in molecules. GraphMem consists of a recurrent controller coupled with an external memory whose cells dynamically interact and change through a multi-hop reasoning process. The dynamic interactions enable an iterative refinement of the representation of molecular graphs with multiple bond types. We demonstrate the effectiveness of the proposed model on 10 BioAssay activity tests.

1 Introduction

Predicting biological activities of molecules in the target environments is a crucial step for virtual screening in the drug discovery pipeline. Much research has focused on the analysis of quantitative structure-activity relationships (QSAR), which results in a myriad of molecular descriptors [3]. For the last 15 years, machine learning has played an important role in the prediction pipeline, that is, mapping the molecular descriptors into its activity classes. Successful machine learning methods are well-established, including kernel methods [2, 8], random forests [16] and gradient boosting [17]. These models take as input a fixed-size feature vector that represents molecular properties, as known as fingerprints. The fingerprint encodes the presence of substructures in a molecule, which are then hashed into a fixed-size feature vector. However, the number of substructures in large molecules might be huge, leading to many hash collisions and information loss.

More recently, *deep learning* [10] has started to impact in drug discovery [1], following their record-breaking successes in vision and languages. The new power comes from a mixture of better architectures (e.g., with hundreds of layers), better training algorithms (e.g., dropout, batch normalization and adaptive gradient descents), and faster tensor–native processors (e.g., graphic processing units). One of the initial successes was the winning of the Merck molecular activity challenge[1] in 2012 by deep neural nets [4]. Another crucial property of deep learning is that it is very flexible in modeling data structures such as images, sequences and graphs. Recently, molecular structures have been successfully modeled using graph convolutional networks [6, 9].

In this paper, we propose Graph Memory Network (GraphMem), a neural architecture that generalizes a powerful recent model known as End-to-End Memory Network [15] and apply it for modeling the graph structure of molecules. The original Memory Network consists of a controller coupled with

[1] https://www.kaggle.com/c/MerckActivity

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

an unstructured and static external memory, organized as an unordered set of cells. The controller reads from the memory in an attentive scheme through multiple reasoning steps before predicting an output. GraphMem, on the other hand, is equipped with *a structured dynamic memory organized as a graph of cells*. The memory cells interact during the reasoning process, and the memory content is refined along the way. The GraphMem is then applied for modeling molecules and predicting its bioactivities as follows. First, raw atom descriptors (or atom embedding) are loaded into memory cells, one atom per cell, and chemical bonds dictate cell connections. A memory cell can recurrently evolve by receiving signals from the controller and the neighbor cells. We validate GraphMem on 10 BioAssay activity tests from the PubChem database[2].

2 Graph Memory Networks

In this section, we briefly present Graph Memory Networks (GraphMem) and show how it can be applied for modeling molecules and predicting bioactivities. An illustration is given in Fig. 1.

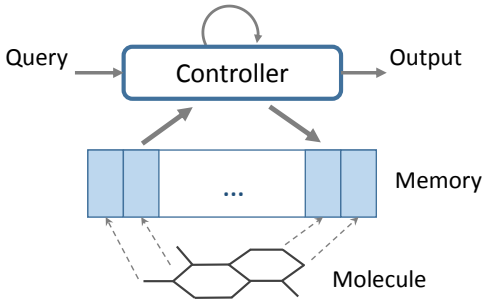

Figure 1: Graph Memory Network that reads a molecule and a query, and generates the query-specific output.

GraphMem consists of a controller and an external memory, both of which are recurrent neural networks (RNNs) interacting with each other. Different from the standard RNNs, the memory is a matrix RNN [5], where the hidden states are matrices with a graph imposed on columns. The controller first takes the query as the input and repeatedly reads from the memory using an attention mechanism, processes and sends the signals back to the memory cells. Each memory cell is updated by the signals from the controller and its neighbor memory cells in the previous step. Through multiple steps of reasoning, the memory cells are evolved from the original input to a refined stage, preparing the controller for generating the output.

The query setting is flexible as it has been demonstrated in question-answering tasks [15]. The query can be any question about different types of activities or properties of a molecule. The question can be embedded by a one-hot vector or an embedding matrix. In the present work, since the task is limited to molecular activity prediction, the query is fixed to a constant vector.

Memory Representation Let M be the number of atoms in the molecule. Each atom i has a feature vector (either pre-computed, or through embedding) $\boldsymbol{x}^i \in \mathbb{R}^{K_x}$. The memory consists of M memory cells, each cell $\boldsymbol{m}_t^i \in \mathbb{R}^{K_m}$ stores the information of the atom i. The memory cells are initialized by a transformation of the feature vectors: $\boldsymbol{m}_1^i = g\left(\boldsymbol{x}^i\right)$. The memory cells connects to each other based on the graph structure of the molecules. If two atoms bond in the molecule, their corresponding memory cells have a connection. This enables the memory cells to embed the substructures of the molecule by updating its content by the information from its neighbors.

Attentive Reading To read from the memory, a content-based addressing scheme, also known as soft attention, is employed. Let $\boldsymbol{h}_t$ be the state of the controller at time t. At each time step t $(t = 1, ..., T)$, the controller reads a summation vector $\boldsymbol{m}_t$ from the memory, which is a sum of all memory cells, weighted by the probability p_t^i, for $i = 1, ...M$:

$$\begin{aligned} \boldsymbol{a}_t^i &= \tanh\left(W_a \boldsymbol{m}_{t-1}^i + U_a \boldsymbol{h}_{t-1}\right) \\ p_t^i &= \text{softmax}\left(\boldsymbol{v}^\top \boldsymbol{a}_t^i\right) \\ \boldsymbol{m}_t &= \sum_i p_t^i \boldsymbol{m}_{t-1}^i \end{aligned}$$

where $\boldsymbol{a}_t^i$ integrates information stored in the memory cell $\boldsymbol{m}_{t-1}^i$ and the controller state $\boldsymbol{h}_{t-1}$, and $\boldsymbol{v}$ is a parameter vector used to measure the contribution of memory cells to the summation vector. All

[2]https://pubchem.ncbi.nlm.nih.gov/

biases are omitted. With this attention mechanism, the controller can selectively choose important atoms toward the predictive output, rather than consider them equally.

Memory updating During the multi-hop reasoning process to answer the query, the controller reads the summation vector $\boldsymbol{m}_t$ from the memory and updates its state as follows:

$$\boldsymbol{h}_t = g\left(W_h \boldsymbol{h}_{t-1} + U_h \boldsymbol{m}_t\right) \tag{1}$$

Each memory cell is updated by the signals from the controller and from the neighbor memory cells in the previous step

$$\boldsymbol{m}_t^i = g\left(W_m \boldsymbol{m}_{t-1}^i + U_m \boldsymbol{h}_t + \sum_r V_r \boldsymbol{c}_{tr}^i\right) \tag{2}$$

$$\boldsymbol{c}_{tr}^i = \frac{1}{|\mathcal{N}_r(i)|} \sum_{j \in \mathcal{N}_r(i)} \boldsymbol{m}_t^j \tag{3}$$

where $\mathcal{N}_r(i)$ is the neighbor atoms of atom i with the bond type r and $\boldsymbol{c}_{tr}^i$ denotes the neighboring context of bond type r. This update allows each memory cell to embed the neighbor information in its representation, thus, capture the graph structure information.

The controller predicts an output after the process of reasoning and updating. The output can be of any type corresponding to the query. In our experiments with molecular activity prediction, the output is either "active" or "inactive".

Recurrent skip-connections Both the controller and the memory updates are implemented using skip-connections [11, 14]

$$\boldsymbol{z}_t = \boldsymbol{\alpha} * \tilde{\boldsymbol{z}}_t + (1 - \boldsymbol{\alpha}) * \boldsymbol{z}_{t-1}$$

where $\boldsymbol{\alpha}$ is a sigmoid gate moderating the amount of information flowing from the previous step, $\boldsymbol{z}_{t-1}$ is the state from the previous step and $\tilde{\boldsymbol{z}}_t$ is a proposal of the new state which is typically implemented as a nonlinear function of $\boldsymbol{z}_{t-1}$.

The controller $\boldsymbol{h}_t$ and the memory cell $\boldsymbol{m}_t^i$ are updated in a fashion similar to that of $\boldsymbol{z}_t$ while $\tilde{\boldsymbol{h}}_t$ and $\tilde{\boldsymbol{m}}_t^i$ are computed as in Eq. (1 and 2). This makes the memory cell update similar to the one in Differential Neural Computer [7], where the memory cells are partially erased and updated with new information.

3 Experiments and Results

Datasets We conducted experiments on 10 **BioAssay** activity tests collected from the PubChem website [3]. Each BioAssay test contains records of activities for chemical compounds. We chose the 2 most common activities for classification: "active" and "inactive". The numbers of molecules in the 10 tests range from 38K to 160K and the numbers of active molecules are from 3K to 60K. Each molecule is represented as a graph, where nodes are atoms and edges are bonds between them.

Baselines The first set of baselines are three common classifiers: SVM, Random Forest (RF) and Gradient Boosting Machine (GMB) on Circular Fingerprint features [12]. Another baseline is Neural Fingerprint (NeuralFP) [6].

Feature extraction For baselines, we use the RDKit toolkit to extraction circular fingerprints[4]. The dimension of the fingerprint features is set by 1024. For our model, RDKit is used to extract the structure of molecules and the atom features. An atom feature vector is the concatenation of the one-hot vector of the atom and other features such as atom degree and number of Hydrogen atoms attached. We also make use of bond features such as bond type and a binary value indicating if it is a bond in a ring.

[3] https://pubchem.ncbi.nlm.nih.gov/

[4] http://www.rdkit.org/

Experiment settings The training minimizes the cross-entropy loss in an end-to-end fashion. We use ReLU units for all steps and Dropout [13] is applied at the first and the last steps of the controller and the memory cells. We set the number of hops by $T = 10$ and other hyper-parameters are tuned on the validation dataset.

Results Table 1 reports results, measured in AUC, on the BioAssay datasets. The proposed GraphMem is competitive against best feature engineering techniques (circular fingerprint and high-performing classifiers). The datasets are listed by the ascending order of dataset sizes.

Dataset	FP+SVM	FP+RF	FP+GBM	NeuralFP	GraphMem
Lung	85.1	85.2	81.5	**85.5**	85.3
Leukemia	82.1	82.1	82.3	**84.5**	84.2
Yeast	77.3	76.5	77.0	79.5	**81.7**
A504333	90.3	90.5	90.6	**90.8**	90.3
A504339	87.4	**87.9**	87.5	**87.9**	87.6
A1814	90.0	89.8	89.6	89.4	**90.1**
A504332	85.0	85.3	85.5	84.3	**85.9**
A2314	88.4	87.9	88.2	86.6	**89.2**
A686979	88.5	87.0	87.9	86.4	**89.3**
A686978	90.1	87.8	89.5	89.3	**90.4**
Average	*86.4*	*86.0*	*86.0*	*86.4*	***87.4***

Table 1: Area under the ROC curve (AUC) (%) for BioAssay datasets. FP = Fingerprint; RF = Random Forests; GBM = Gradient Boosting Machine

Fig. 2 reports the F1-score on the 10 datasets. On average, GraphMem beats all the baselines.

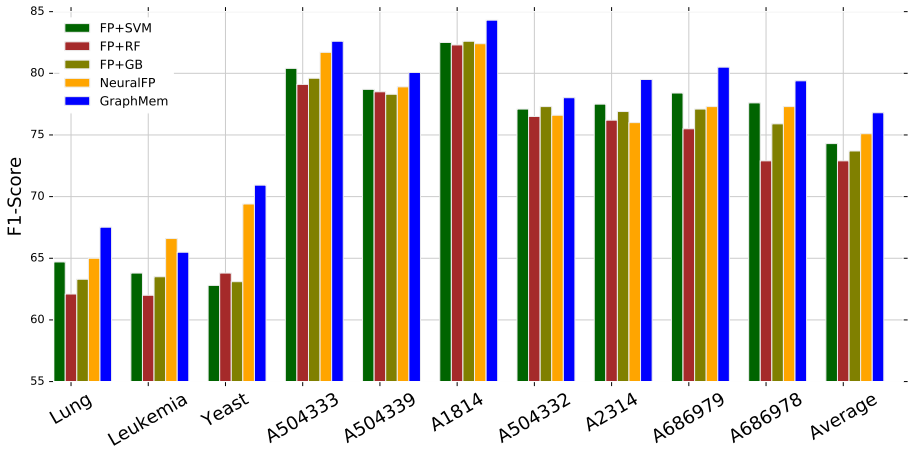

Figure 2: F1-score (%) for NCI datasets. FP = Fingerprint; RF = Random Forests; GBM = Gradient Boosting Machine. Best view in color.

4 Discussion

We have proposed Graph Memory Network (GraphMem), a neural network augmented with a dynamic and graph-structured memory and applied it for modeling molecules. Experiments on 10 BioAssay activity tests demonstrated that GraphMem is effective in answering queries about bioactivities of large molecules given only the molecular graphs.

There is room for further investigations. First, BioAssay activity ground truths used in training for each target (e.g., a disease) are expensive to establish. We can leverage the strength of statistics from the existing large datasets to improve over the smaller datasets. For example, each BioAssay test can be considered as a task and the model can jointly learn all tasks. The task ID and other information of the molecule can be embedded in the query. Second, the memory structure in GraphMem, once constructed from data graphs, is then fixed even though the content of the memory changes during the reasoning process. A future work would be deriving dynamic memory graphs that evolve with time.

References

[1] Igor I Baskin, David Winkler, and Igor V Tetko. A renaissance of neural networks in drug discovery. *Expert opinion on drug discovery*, 11(8):785–795, 2016.

[2] Robert Burbidge, Matthew Trotter, B Buxton, and Sl Holden. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers & chemistry*, 26(1):5–14, 2001.

[3] Artem Cherkasov, Eugene N Muratov, Denis Fourches, Alexandre Varnek, Igor I Baskin, Mark Cronin, John Dearden, Paola Gramatica, Yvonne C Martin, Roberto Todeschini, et al. QSAR modeling: where have you been? Where are you going to? *Journal of medicinal chemistry*, 57(12):4977–5010, 2014.

[4] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[5] Kien Do, Truyen Tran, and Svetha Venkatesh. Learning recurrent matrix representation. *Third Representation Learning for Graphs Workshop (ReLiG 2017)*, 2017.

[6] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[7] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[8] Robert N Jorissen and Michael K Gilson. Virtual screening of molecular databases using a support vector machine. *Journal of chemical information and modeling*, 45(3):549–561, 2005.

[9] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[11] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Faster training of very deep networks via p-norm gates. *ICPR*, 2016.

[12] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[14] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[15] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *NIPS*, 2015.

[16] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of chemical information and computer sciences*, 43(6):1947–1958, 2003.

[17] Vladimir Svetnik, Ting Wang, Christopher Tong, Andy Liaw, Robert P Sheridan, and Qinghua Song. Boosting: An ensemble learning tool for compound classification and qsar modeling. *Journal of chemical information and modeling*, 45(3):786–799, 2005.