Graph convolutional Fingerprints

D. Duvenaud, D. Maclaurin *et al*, NIPS *conf.*, 2018.

MODNBIO

Outline

- Molecule graph convolution (MGC) 의 개요
  - Graph convolutional FP의 장점
- Graph convolutional FP algorithm
- Results
- Follow-up study

Graph convolutional fingerprints 개요

- Graph convolutional fingerprints (GCFP) 의 장점
  - Predictive performance
    - GCFP는 용해성(solubility), 약효검정(drug efficacy) 등에서 표준 FP 보다 좋은 성능을 나타낸다.
  - Parsimony
    - fixed length의 FP는 molecule의 다양한 구조를 포함하기 위해 매우 큰 digit을 가짐. 그에 반해 GCFP는 필요한 digit을 절약한다.
  - Interpretability
    - 표준 FP는 molecule의 각각의 하부구조를 구별해서 생성되는데, neural graph FP는 하부구조의 유사성과 구별성을 고려하기 때문에 더 의미가 있는 FP를 만들 수 있다.

Graph convolutional fingerprints algorithm

Algorithm 1 Circular fingerprints

Input: molecule, radius R, fingerprint length S
Initialize: fingerprint vector $\mathrm{f} \leftarrow 0_S$
for each atom a in molecule
  $\mathrm{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule
    $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
    $\mathrm{v} \leftarrow [\mathrm{r}_a, \mathrm{r}_1, \ldots, \mathrm{r}_N]$ ▷ concatenate
    $\mathrm{r}_a \leftarrow \mathrm{hash}(\mathrm{v})$ ▷ hash function
    $i \leftarrow \mathrm{mod}(r_a, S)$ ▷ convert to index
    $\mathrm{f}_i \leftarrow 1$ ▷ Write 1 at index
Return: binary vector f

Algorithm 2 Neural graph fingerprints

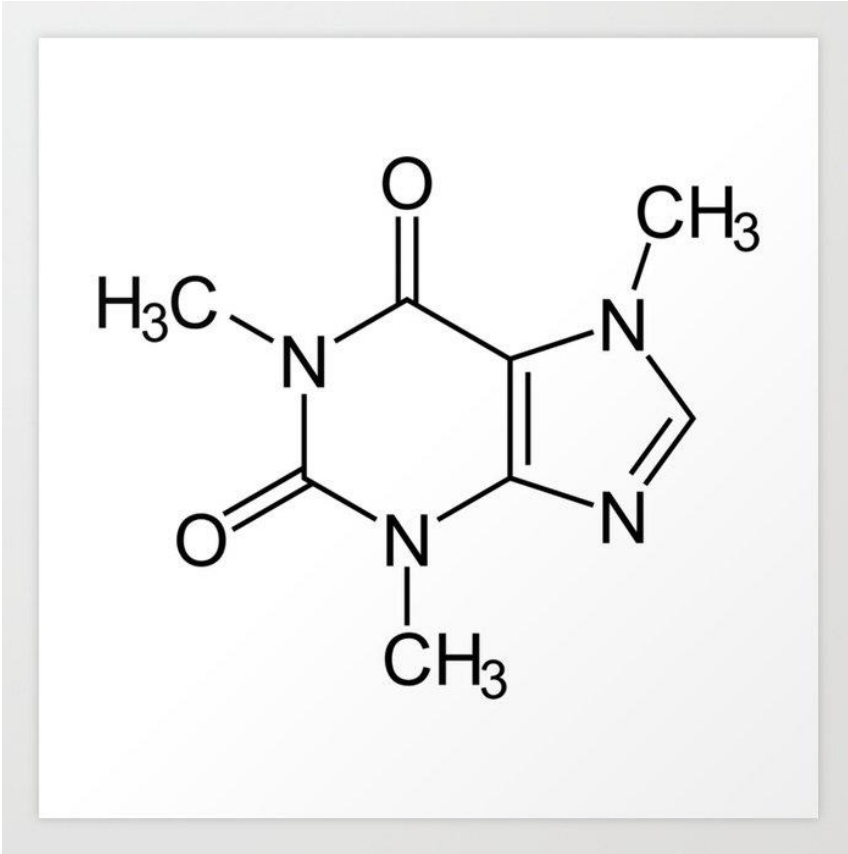
Input: molecule, radius R, hidden weights $H_1^1 \ldots H_R^5$, output weights $W_1 \ldots W_R$
Initialize: fingerprint vector $\mathrm{f} \leftarrow 0_S$
for each atom a in molecule
  $\mathrm{r}_a \leftarrow g(a)$ ▷ lookup atom features
for $L = 1$ to R ▷ for each layer
  for each atom a in molecule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

Graph convolutional fingerprints algorithm

- 준비물
  - molecule
  - radius R
  - hidden weights H
  - output weights W
  - zero initialized fingerprints $f = [0,0,0,0,0,0,0,0,\ldots,0]$

Algorithm 2 Neural graph fingerprints

1: **Input:** molecule, radius R, hidden weights $H_1^1 \ldots H_R^5$, output weights $W_1 \ldots W_R$
2: **Initialize:** fingerprint vector f $\leftarrow 0_S$

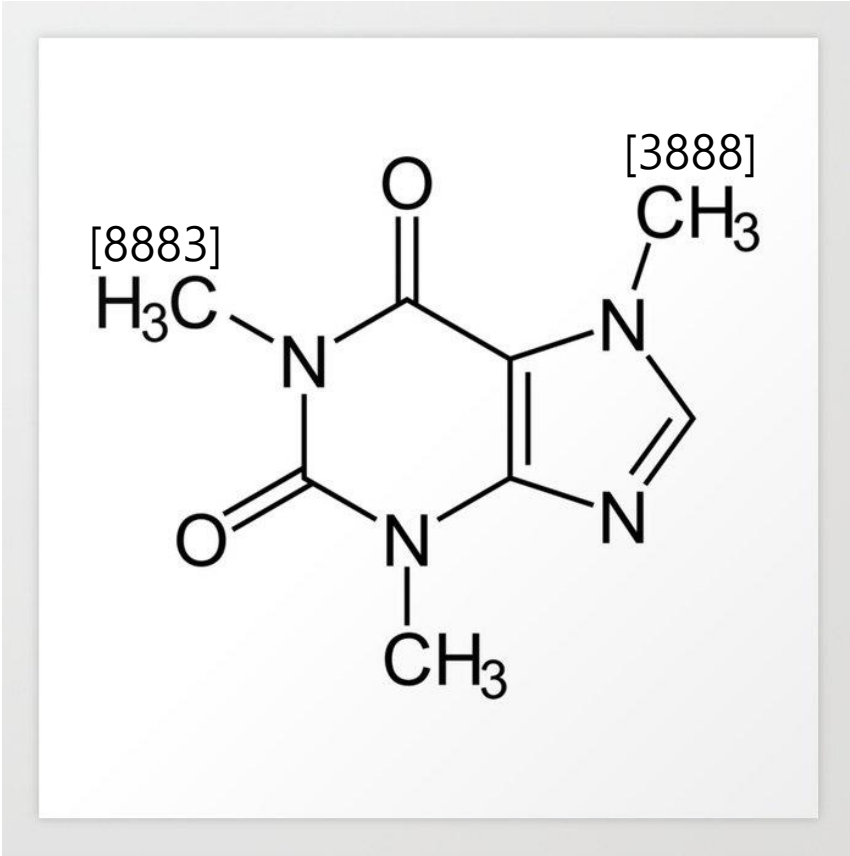


Graph convolutional fingerprints algorithm

- atom의 feature를 살펴본다.
  - atom의 feature란 정보를 의미한다.
  - atom의 feature를 다양한 방식으로 encoding
  - H3C = [8,8,8,3] / N = [14,0,0,0]

$r =$ (14,4)

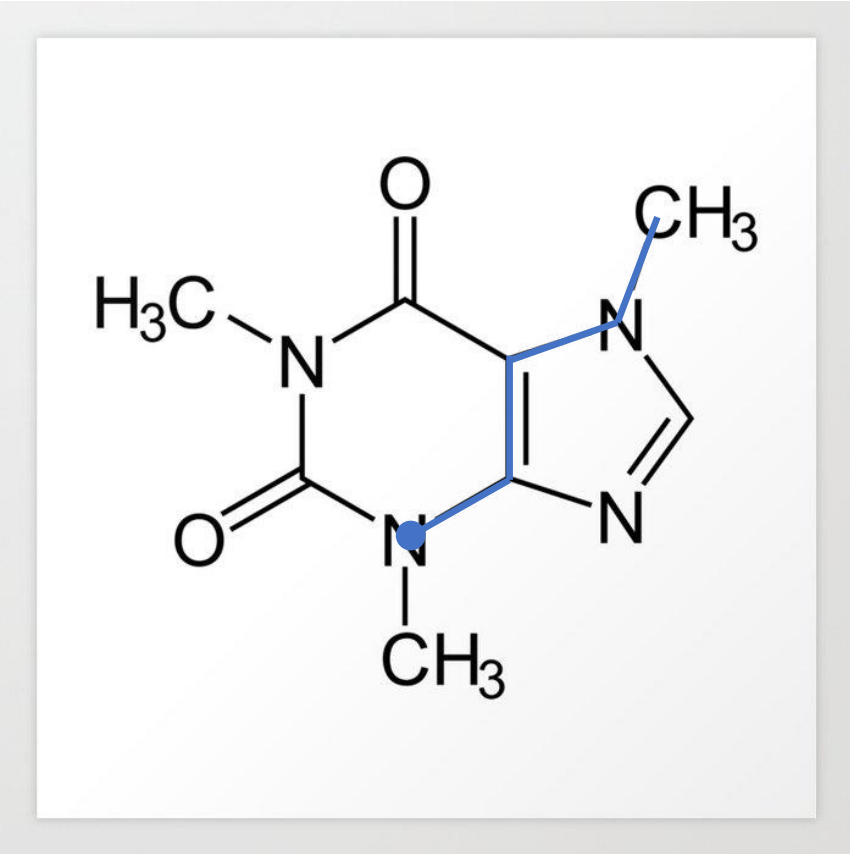
3: **for** each atom a in molecule
4: $\mathrm{r}_a \leftarrow g(a)$ ▷ lookup atom features



Graph convolutional fingerprints algorithm

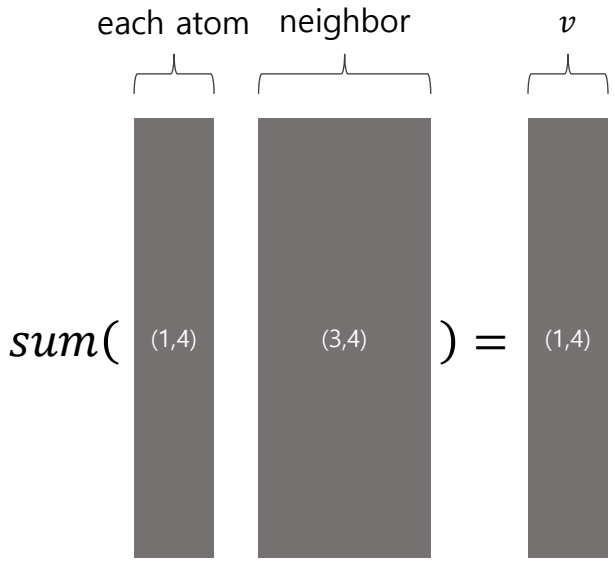
- 1부터 radius R까지 순차적으로 L에 할당
- L iterator = [1,2,3,4]

5: **for** $L = 1$ to R ▷ for each layer

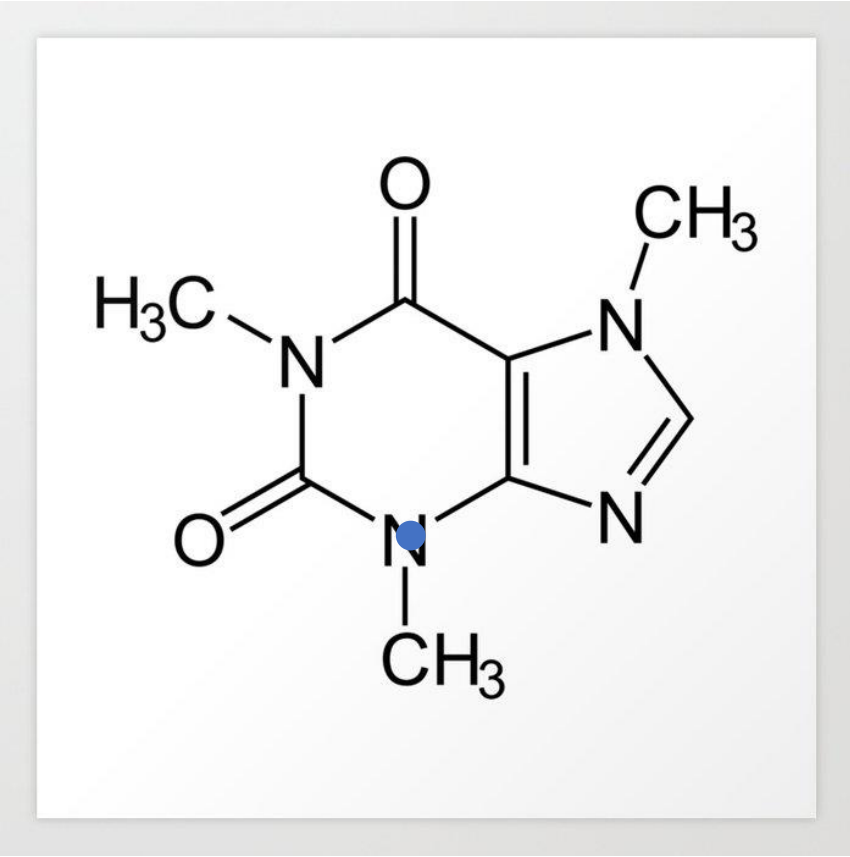


Graph convolutional fingerprints algorithm

- 각 atom에 대해서 모두 아래의 연산을 실행한다.

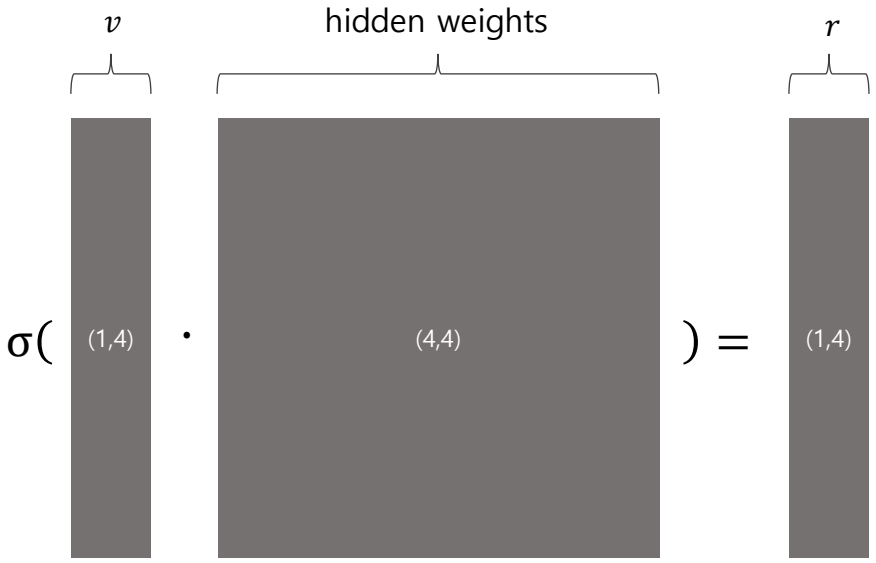


6: **for** each atom a in molecule
7: $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
8: $\mathrm{v} \leftarrow \mathrm{r}_a + \sum_{i=1}^{N} \mathrm{r}_i$ ▷ sum
9: $\mathrm{r}_a \leftarrow \sigma(\mathrm{v}H_L^N)$ ▷ smooth function
10: $\mathrm{i} \leftarrow \mathrm{softmax}(\mathrm{r}_a W_L)$ ▷ sparsify
11: $\mathrm{f} \leftarrow \mathrm{f} + \mathrm{i}$ ▷ add to fingerprint

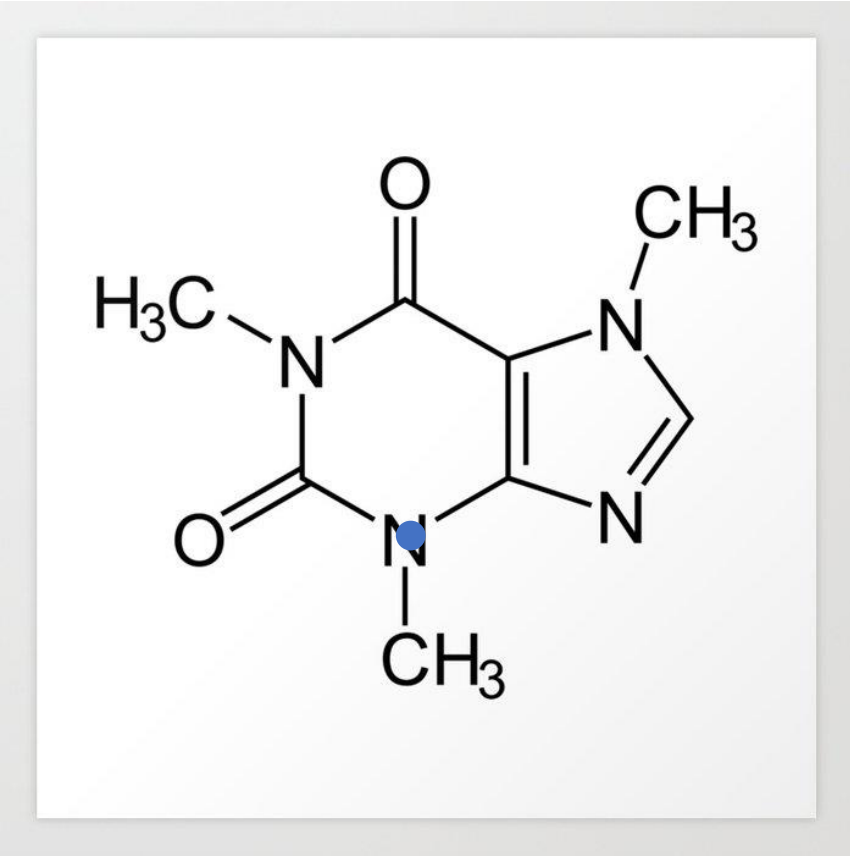


Graph convolutional fingerprints algorithm

- 각 atom에 대해서 모두 아래의 연산을 실행한다.

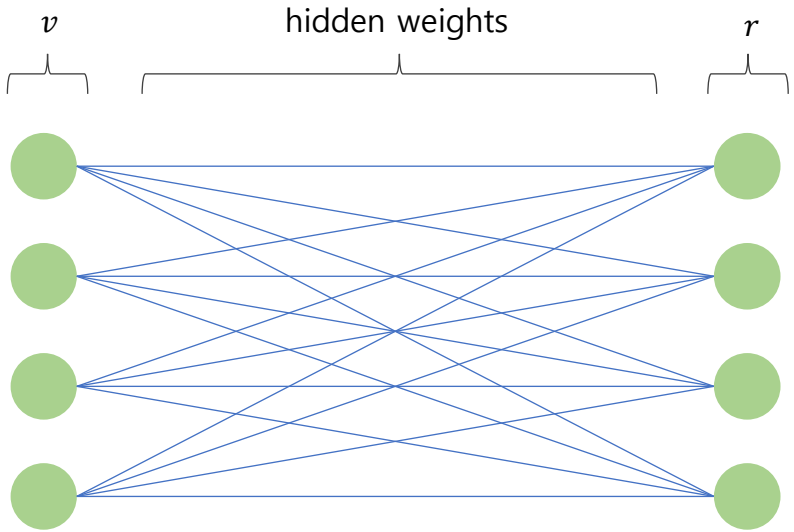


for each atom a in molecule
  $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
  $\mathrm{v} \leftarrow \mathrm{r}_a + \sum_{i=1}^{N} \mathrm{r}_i$ ▷ sum
  $\mathrm{r}_a \leftarrow \sigma(\mathrm{v} H_L^N)$ ▷ smooth function
  $\mathrm{i} \leftarrow \mathrm{softmax}(\mathrm{r}_a W_L)$ ▷ sparsify
  $\mathrm{f} \leftarrow \mathrm{f} + \mathrm{i}$ ▷ add to fingerprint

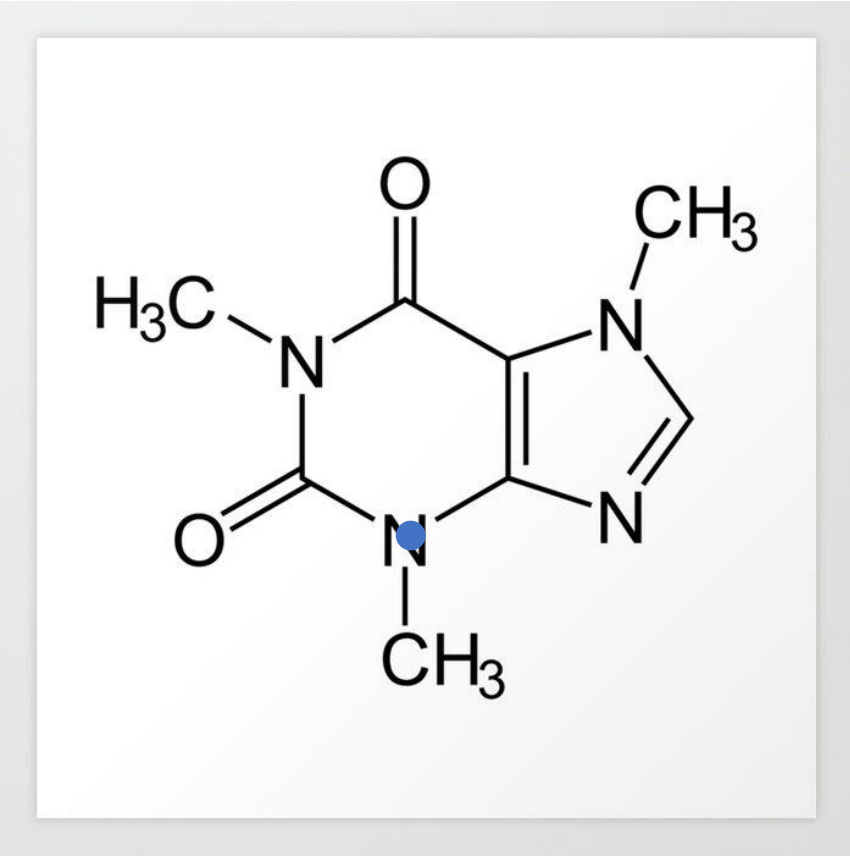


Graph convolutional fingerprints algorithm

- 각 atom에 대해서 모두 아래의 연산을 실행한다.

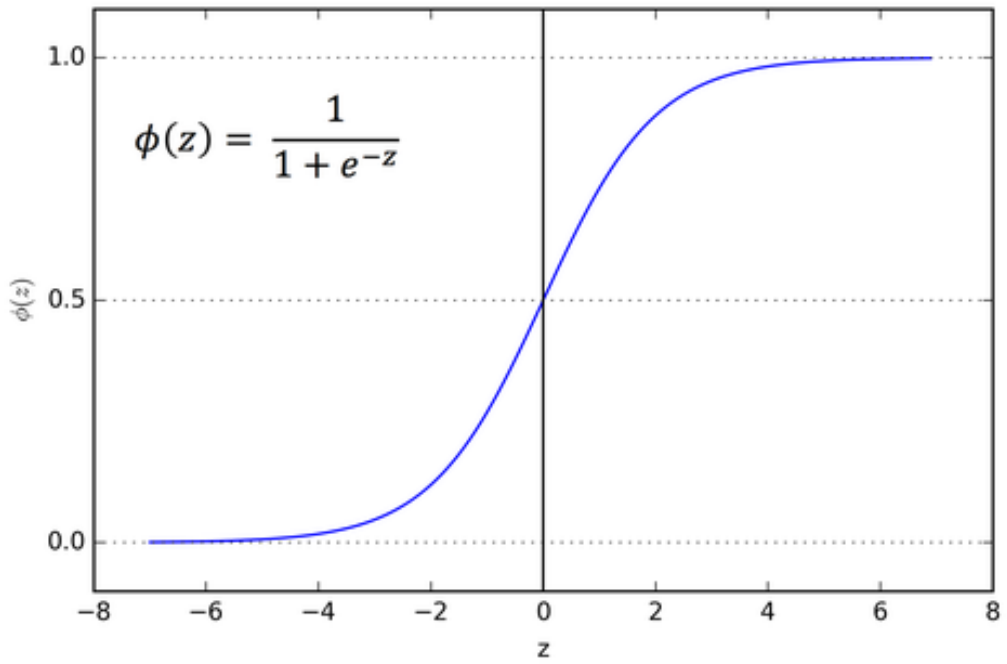


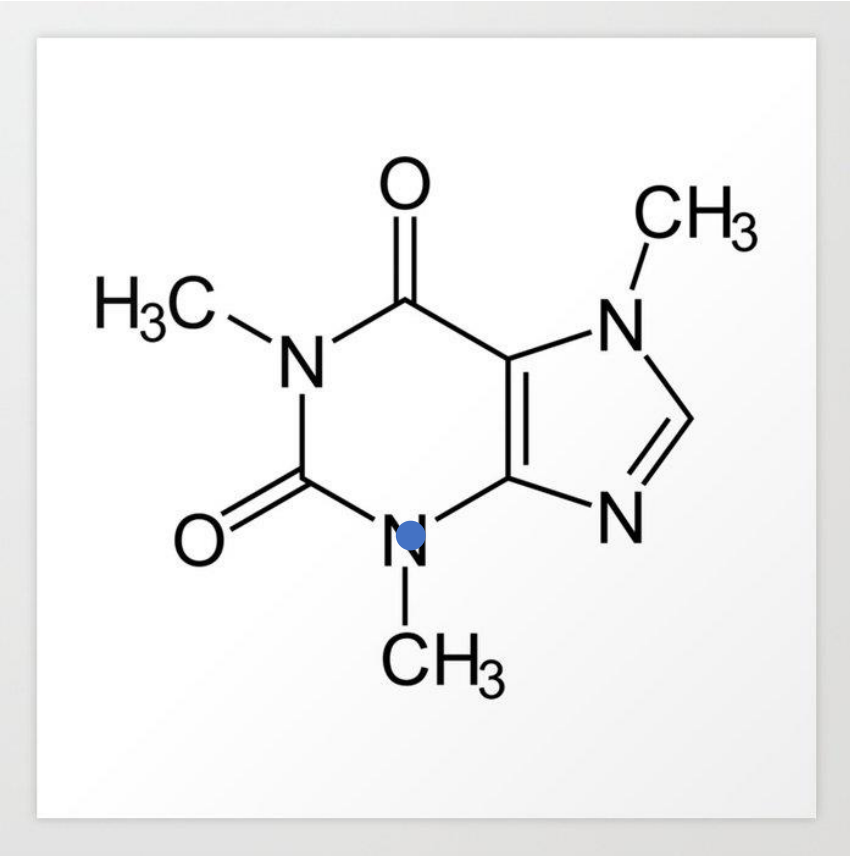
for each atom a in molecule
  $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
  $\mathrm{v} \leftarrow \mathrm{r}_a + \sum_{i=1}^{N} \mathrm{r}_i$ ▷ sum
  $\mathrm{r}_a \leftarrow \sigma(\mathrm{v}H_L^N)$ ▷ smooth function
  $\mathrm{i} \leftarrow \mathrm{softmax}(\mathrm{r}_a W_L)$ ▷ sparsify
  $\mathrm{f} \leftarrow \mathrm{f} + \mathrm{i}$ ▷ add to fingerprint



Graph convolutional fingerprints algorithm

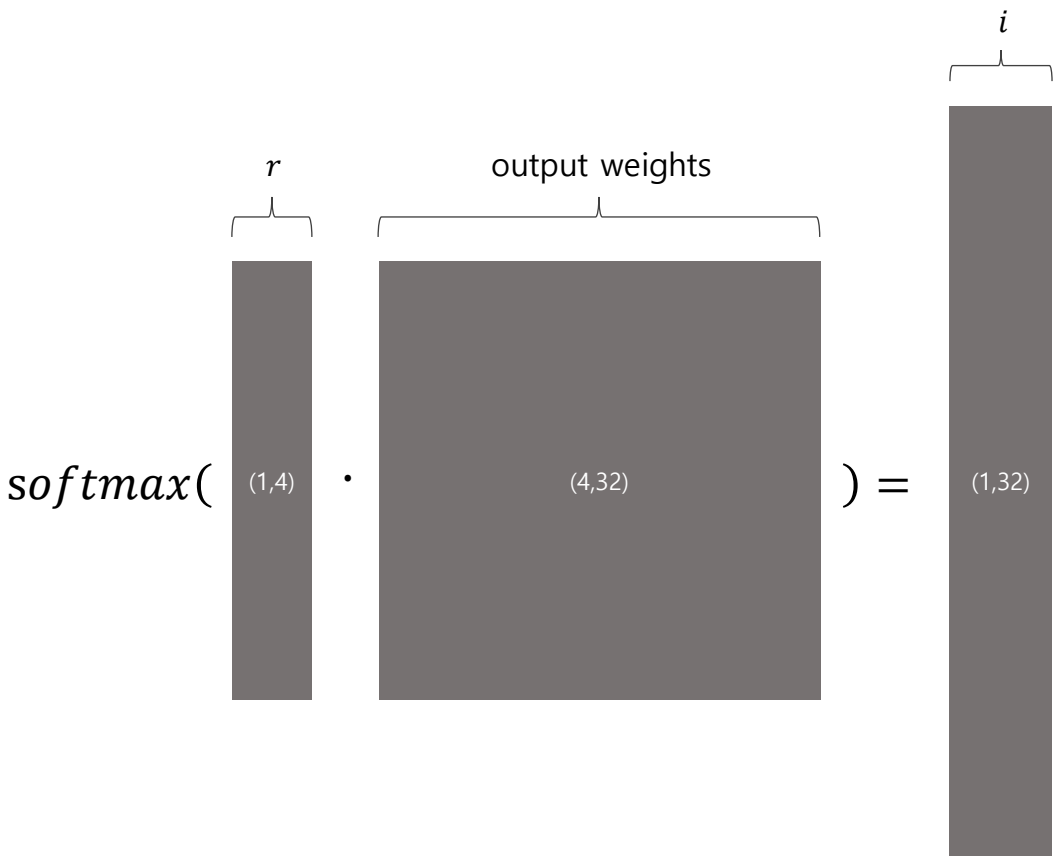
- 각 atom에 대해서 모두 아래의 연산을 실행한다.

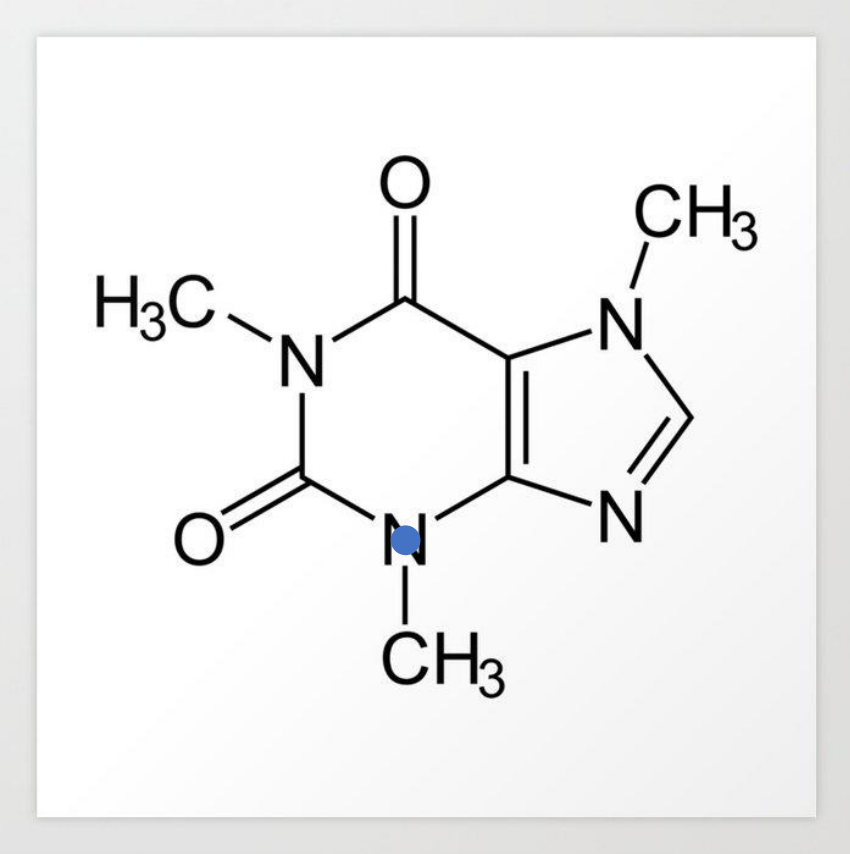




Graph convolutional fingerprints algorithm

- 각 atom에 대해서 모두 아래의 연산을 실행한다.

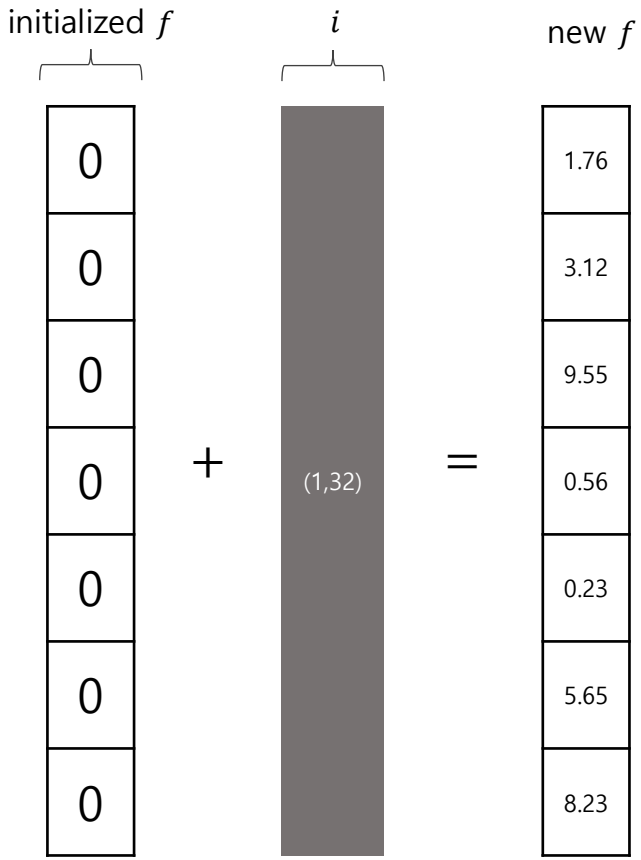


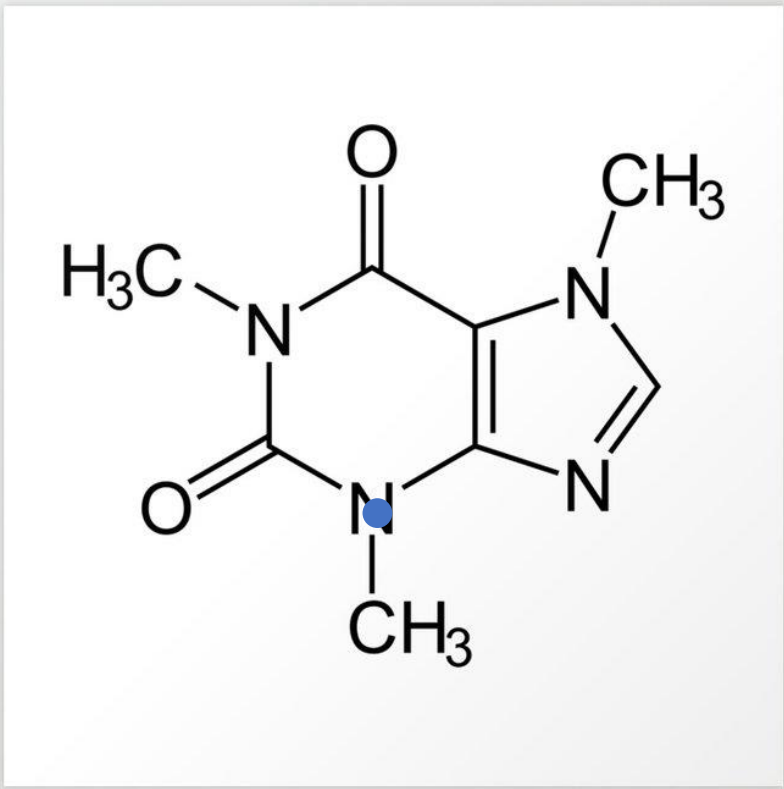


for each atom a in molecule
  $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
  $\mathrm{v} \leftarrow \mathrm{r}_a + \sum_{i=1}^{N} \mathrm{r}_i$ ▷ sum
  $\mathrm{r}_a \leftarrow \sigma(\mathrm{v}H_L^N)$ ▷ smooth function
  $\mathrm{i} \leftarrow \mathrm{softmax}(\mathrm{r}_a W_L)$ ▷ sparsify
  $\mathrm{f} \leftarrow \mathrm{f} + \mathrm{i}$ ▷ add to fingerprint

Graph convolutional fingerprints algorithm

- 각 atom에 대해서 모두 아래의 연산을 실행한다.





6: **for** each atom a in molecule
7: $\mathrm{r}_1 \ldots \mathrm{r}_N = \mathrm{neighbors}(a)$
8: $\mathrm{v} \leftarrow \mathrm{r}_a + \sum_{i=1}^{N} \mathrm{r}_i$ ▷ sum
9: $\mathrm{r}_a \leftarrow \sigma(\mathrm{v}H_L^N)$ ▷ smooth function
10: $\mathrm{i} \leftarrow \mathrm{softmax}(\mathrm{r}_a W_L)$ ▷ sparsify
11: $\mathrm{f} \leftarrow \mathrm{f} + \mathrm{i}$ ▷ add to fingerprint

Results

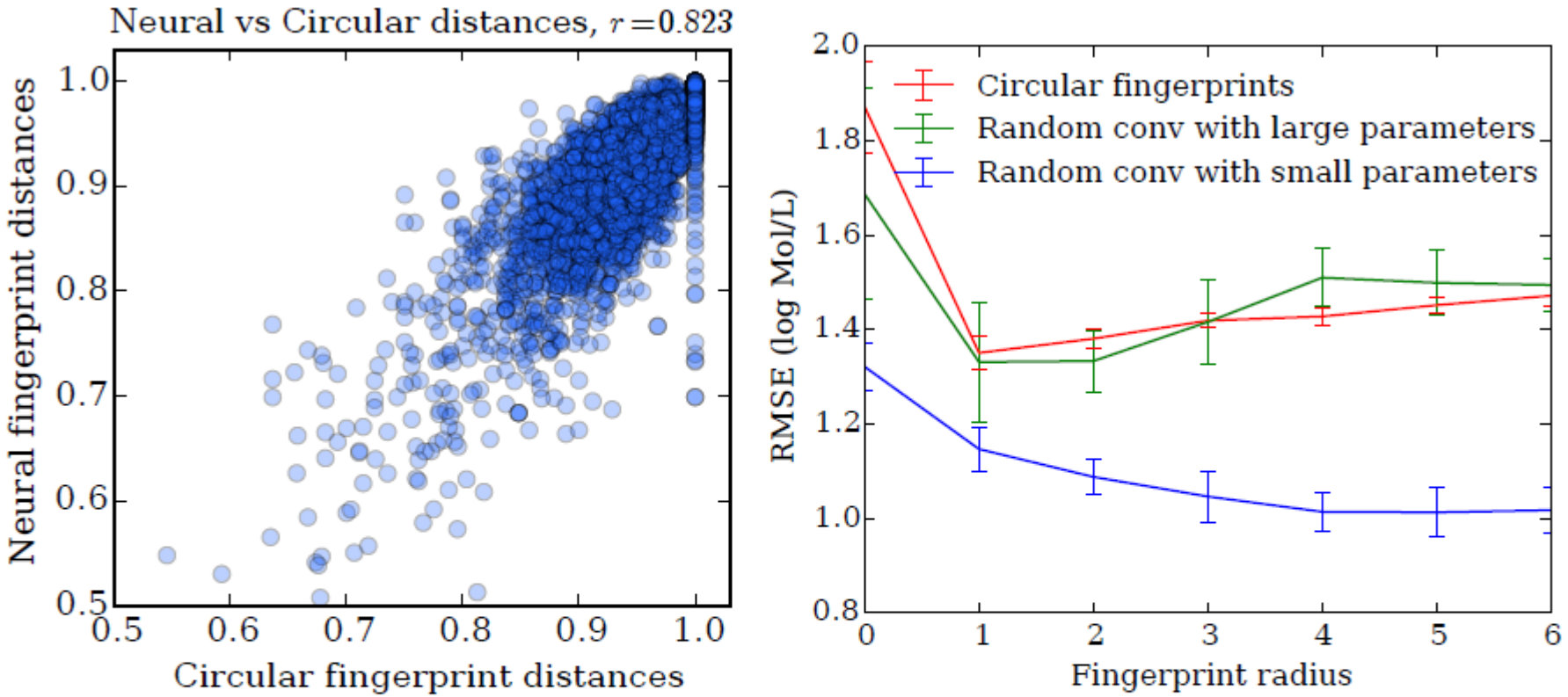


Figure 3: *Left:* Comparison of pairwise distances between molecules, measured using circular fingerprints and neural graph fingerprints with large random weights. *Right*: Predictive performance of circular fingerprints (red), neural graph fingerprints with fixed large random weights (green) and neural graph fingerprints with fixed small random weights (blue). The performance of neural graph fingerprints with large random weights closely matches the performance of circular fingerprints.

Follow-up study