Learning Molecular Fingerprints from the Graph Up

David Duvenaud, Dougal Maclaurin,

Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli,

Timothy Hirzel, Alán Aspuru-Guzik, Ryan P. Adams
Motivation

• Want to do regression on molecules
• For virtual screening of drugs, materials, etc.
• Problem: Molecules can be any size and shape
• Only know how to learn from fixed-size examples.
• How to take a molecule in and produce a fixed-size vector?
Circular Fingerprints

- Standard method lists all substructures below a certain size
- Can do this by combining hashes of each atom with and bonded neighbors
- Hash value indexes into a fixed-sized vector
- Problem: can’t optimize with gradients
What would Ryan do?

- Maybe we can build a message-passing network
- same function is applied to each node (atom) and its neighbors
- Like a convolutional net
- At the top, add all node’s vectors together
- If we use a softmax, this generalizes circular fingerprints
Continuous-izing Circular Fingerprints

Circular fingerprints

1: **Input:** molecule, radius $R$, fingerprint length $S$
2: **Initialize:** fingerprint vector $f \leftarrow 0_S$
3: for each atom $a$ in molecule do
4: $r_a \leftarrow g(a)$ \(\triangleright\) lookup atom features
5: for $L = 1$ to $R$ do \(\triangleright\) for each layer
6: for each atom $a$ in molecule do
7: $r_1 \ldots r_N = \text{neighbors}(a)$
8: $v \leftarrow [r_a, r_1, \ldots, r_N]$ \(\triangleright\) concatenate
9: $r_a \leftarrow \text{hash}(v)$ \(\triangleright\) hash function
10: $i \leftarrow \text{mod}(r_a, S)$ \(\triangleright\) convert to index
11: $f_i \leftarrow 1$ \(\triangleright\) Write $1$ at index
12: **Return:** binary vector $f$

Neural graph fingerprints

1: **Input:** molecule, radius $R$, weights $H_1 \ldots H_R$, output weights $W_1 \ldots W_R$
2: **Initialize:** fingerprint vector $f \leftarrow 0_S$
3: for each atom $a$ in molecule do
4: $r_a \leftarrow g(a)$ \(\triangleright\) lookup atom features
5: for $L = 1$ to $R$ do \(\triangleright\) for each layer
6: for each atom $a$ in molecule do
7: $r_1 \ldots r_N = \text{neighbors}(a)$
8: $v \leftarrow r_a + \sum_{i=1}^{N} r_i$ \(\triangleright\) sum
9: $r_a \leftarrow \sigma(v H^N_L)$ \(\triangleright\) smooth function
10: $i \leftarrow \text{softmax}(r_a W_L)$ \(\triangleright\) sparsify
11: $f \leftarrow f + i$ \(\triangleright\) add to fingerprint
12: **Return:** real-valued vector $f$

Every non-differentiable operation is replaced with a differentiable analog.
Generalizing Circular Fingerprints

- If we generalize existing fingerprints, we can’t not win (unless we overfit)
- Large random weights makes neural nets act like hash functions
- Looked at similarities between pairwise distances.

Neural vs Circular distances, $r = 0.823$
Generalizing Circular Fingerprints

- If we generalize existing fingerprints, we can’t not win (unless we overfit)
- Large random weights makes neural nets act like hash functions
- Looked at performance of random weights.
### Performance

<table>
<thead>
<tr>
<th>Dataset Units</th>
<th>Solubility log Mol/L</th>
<th>Drug efficacy EC&lt;sub&gt;50&lt;/sub&gt; in nM</th>
<th>Photovoltaic efficiency percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predict mean</td>
<td>4.29 ± 0.40</td>
<td>1.47 ± 0.07</td>
<td>6.40 ± 0.09</td>
</tr>
<tr>
<td>Circular FPs + linear layer</td>
<td>1.84 ± 0.08</td>
<td>1.13 ± 0.03</td>
<td>2.62 ± 0.07</td>
</tr>
<tr>
<td>Circular FPs + neural net</td>
<td>1.40 ± 0.15</td>
<td>1.24 ± 0.03</td>
<td>2.04 ± 0.07</td>
</tr>
<tr>
<td>Neural FPs + linear layer</td>
<td>0.74 ± 0.09</td>
<td>1.16 ± 0.03</td>
<td>2.71 ± 0.13</td>
</tr>
<tr>
<td>Neural FPs + neural net</td>
<td>0.53 ± 0.07</td>
<td>1.17 ± 0.03</td>
<td>1.44 ± 0.11</td>
</tr>
</tbody>
</table>

- Could also try varying depth of neural net on top (used one hidden layer here)
Interpretability

- Circular fingerprints activate for a single substructure
- No generalization
- No notion of similarity
- Let’s put a linear layer on top of neural fingerprints and examine which fragments activate most predictive features.
Interpretability: Solubility

Fragments activating feature most predictive of solubility:

most predictive of insolubility:
Interpretability: Toxicity

Fragments most activated by toxicity feature on SR-MMP dataset:

Fragments most activated by toxicity feature on NR-AHR dataset:
Future Work

• Limitation: Slow because of so many weight transforms
• Could use low-rank weight matrices
• Limitation: All features are local
• Could learn to “parse” molecules
• But how to take gradients?
