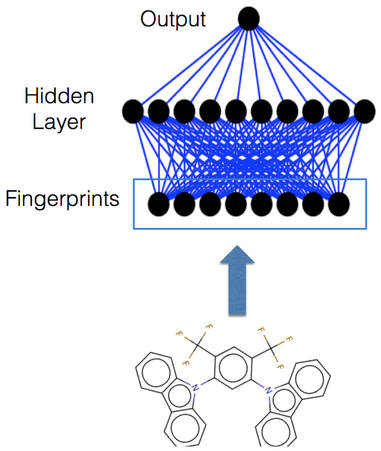
Convolutional Networks on Graphs for Learning Molecular Fingerprints

David Duvenaud*, Dougal Maclaurin*, Jorge Aguilera-Iparraguirre
Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, Ryan P. Adams

HARVARD INTELLIGENT PROBABILISTIC SYSTEMS

Problem

- How to do regression when input is a graph?
- Graphs can be any size or shape.
- Hard to turn into fixed-length vector.
- Molecules are graphs!
- Australians are *it*



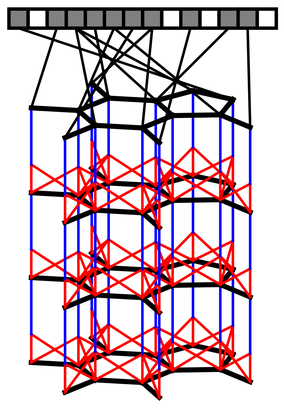
Circular fingerprints

Aka Morgan fingerprints, or ECFP

- Map variable-sized molecular graph to fixed-length binary vector
- Binary features indicate presence of substructures

Can be efficiently computed using local operations:

- At each layer, hash the features of each atom and its neighbors/bonds
- More layers correspond to increasing radius of substructures
- At top level, interpret as (modulo) integer and set the entry to one



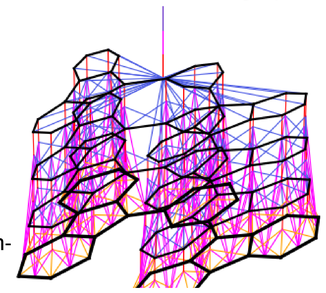
Convolutional neural nets on graphs

How to make these differentiable?

Hash $\rightarrow$ neural net
Index $\rightarrow$ softmax
Write $\rightarrow$ Add

Results in an end-to-end differentiable convolutional network

Information flow graph

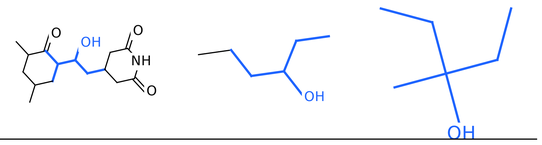


Message passing between neighbors, plus final pooling step

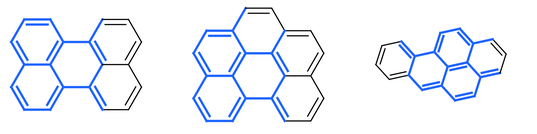
Neural fingerprints are interpretable

When trained with linear layer on top, can see which fragments most contribute to prediction

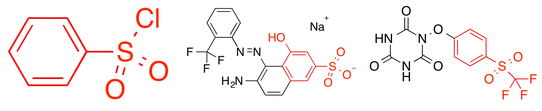
Fragments most activated by pro-solubility feature



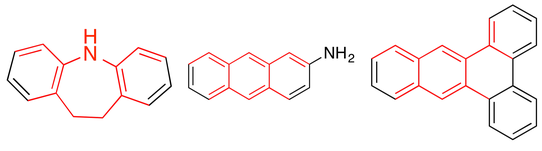
Fragments most activated by anti-solubility feature



Fragments most activated by toxicity feature on SR-MMP dataset

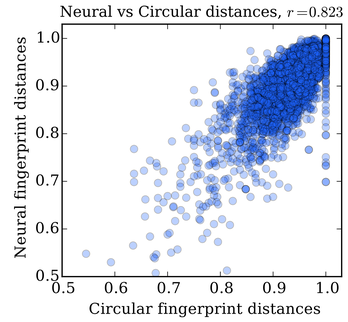


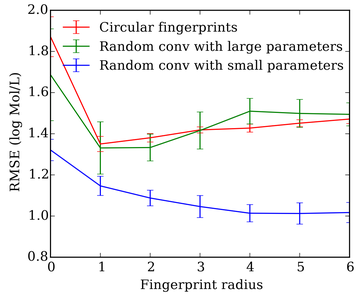
Fragments most activated by toxicity feature on NR-AHR dataset



Neural graph fingerprints generalize circular fingerprints

With large random weights, neural graph fingerprints approach circular fingeprints:





Results

Because neural graph fingerprints generalize state of the art, we can't not win:

Dataset	Solubility	Drug efficacy	Photovoltaic efficien
Units	log Mol/L	EC_{50} in nM	percent
Predict mean	2.07 ± 0.10	1.21 ± 0.03	2.53 ± 0.02
Circular FPs + linear layer	1.31 ± 0.05	**1.06 ± 0.01**	1.62 ± 0.03
Circular FPs + neural net	1.18 ± 0.05	1.16 ± 0.04	1.41 ± 0.03
Neural FPs + linear layer	0.87 ± 0.06	**1.07 ± 0.01**	1.61 ± 0.06
Neural FPs + neural net	**0.72 ± 0.05**	**1.08 ± 0.01**	**1.20 ± 0.04**

Mean predictive accuracy of neural fingerprints compared to standard circular fingerprints.

Conclusion

- We can compute gradients of learning procedures...
- This lets us optimize thousands of hyperparameters!
- All code for experiments at `github.com/HIPS/hypergrad`
- We also wrote an autodiff package that works on standard Numpy code: `github.com/HIPS/autograd`