Team: Augur (Kelley Paskov, Greg McInnes, Christine Tataru, and Nate Stockham)

Review of:
Kearnes, S., McCloskey, K. , Berndl, M. ,Pande V., Riley P. (2016). Molecular Graph Convolutions: Moving Beyond Fingerprints. J Comput Aided Mol Des.
https://arxiv.org/abs/1603.00856

Intro:
Traditionally, chemoinformatics and drug discovery applications of machine learning have been applied to "molecular fingerprints", encoded summaries of molecular qualities rather than pure molecular representations. This paper seeks to use deep learning on representations of molecules as undirected graphs, where every atom is a node and every bond is an edge with certain properties. They hope that enabling use of these raw molecular representations will enable creation of less application-specific machine learning models with less bias and more flexibility.

Model
The model takes in a basic representation of a molecule based on a simple molecular graph[(1)] and outputs a vector representing the molecule in n dimensional space. The model is made of two basic units of representation. The first is the atom layer, which contains an n-dimensional vector associated with each atom. The second is a pair layer, a 3 dimensional matrix where the first two dimensions are indexed by atoms. All operations in the model maintain the following 3 types of invariance:

1. Order invariance, which ensures that the output is not dependent on the order of input atoms
2. Atom and pair permutation invariance, which ensures that from a single atom or pair's perspective, it's value in relation to all other atoms and pairs is invariant to the order of atoms or pairs
3. Pair order invariance, which means the value between atom a and b is the same as the value between b and a.

The authors define "Weave Modules", modules that take in an atom matrix and a pair matrix and interrelate them to inform atoms about their chemical environment and form new, modified, atom and pair matrices. These can then be used as inputs into the next weave module. (See more on Weave modules below [(2)])

In order to reduce the many dimensional top-level atom representation output by the weave modules into a single molecular representation, the authors used fuzzy histograms. A fuzzy histogram is defined by a set of bins and membership functions (in this case unnormalized Gaussians), where function i reports the membership of the input point to bin i. The molecular representation, then, is the concatenation of the histograms obtained by applying fuzzy histograms to top-level atom feature vectors using 11 bins spanning a Gaussian distribution of mean 0 and standard deviation 1.

The overall model architecture then consists of some number of weave modules, followed by a final convolution that outputs just an atom layer. There is then a fully connected layer that feeds into a softmax layer, which is customized based on the specific prediction task. Note many models with varying numbers of Weave Modules were tested.

Features:

This method is meant to move away from description of a molecular as an amalgamation of molecular qualities and instead represent a molecule as a collection of atoms and bonds, each with its own properties. As such, each atom is represented as a vector indicating features like: atom type, chirality, charge, aromaticity, etc. Each bond is represented as a vector of bond types, ring structure, etc.

Datasets:

The authors use 259 datasets from 4 sources, PubChem BioAssay, the “maximum unbiased validation(MUV) datasets, the “enhanced directory of useful decoys” (DUD-E), and the training set from the Tox21 challenge.

Training and Validation:

The authors trained their model based on how well standard softmax classification was able to predict bioactivity of input molecule using the molecular representation provided by molecular graph convolutions. They then compared their results to standard and top of the line machine learning classifiers logistic regression, random forests, Max Sim, and pyramidal multitask neural network(PMTNN) baselines, all of which took Morgan molecular fingerprints as inputs. They relied on median AUC and 95% Wilson score interval for a sign test estimating the probability that the model would outperform the PMTNN baseline. Many of the molecular graph convolution models were able to perform just as well as the baseline models, despite the fact that the input was considerably less complex, lacking the hand-picked features of standard molecular fingerprinting.

Notably, a simplified atom and bond featurization, just atom type, bond type, and graph distance, achieved similar scores to the more complex one described above. This suggests the more complex features are either not relevant to the training algorithm, or can be derived from the simplified version.

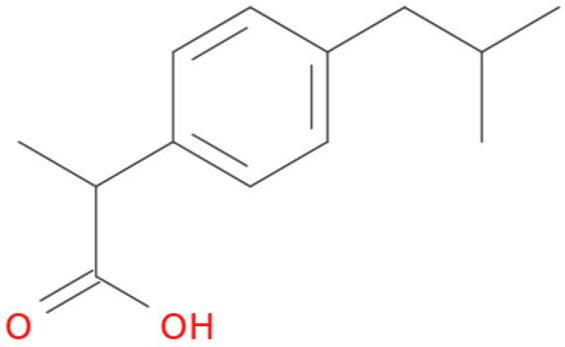
Discussion:

Molecular graph convolution is presented as a proof of concept method to verify that the more raw, descriptive representation of a molecule as atoms, bonds, and pairwise relationships can perform as well as the state of the art multitask neural nets trained on traditional molecular fingerprints. Theoretically the model presents an enormous amount of flexibility, because every molecular representations would no longer be tethered to a featurized, biased vector. A plethora of possible future directions include fine-tuning model hyperparameters like number of Weave modules and exploring other options, apart from fuzzy histograms, of congregating the Weave

Module atom output into a single molecular representation. Additionally, more experiment involving different machine learning classifiers that take the new molecular representation as input could prove fruitful.

Appendix:

1. **Molecular representation of ibuprofen**



2. Weave modules

The first layer, the atom layer, is a two dimensional matrix associating a n-dimensional feature vector with each atom. Note, the features used for the main results are very “simple” in this paper, consisting only of elements that can be directly derived from the 2-d molecular graph.

The second layer in each block is the pair layer, which is a 3-dimensional matrix that encodes an n-dimensional matrix for each atom pair. Note, the atom pair have to have the same index ordering as the atom layer.

With these two layers defined, they develop a set of operations that preserve the above invariants and that can generate a new atom or pair layer from a given atom or pair layer. These operations are named (A->A), (A->P), (P->P), and (P->A). Thus, successive modules can be stacked arbitrarily using these operations. To do this, they use two classes of functions, where “f” is an arbitrary function and “g” is a commutative function (returning the same result regardless of the order of the dimensions).

The first operations they define, (A->A) and (P->P), use “f” to combine the values of a given atom “a” or a given pair “(a,b)” across layers. By using as input to “f” the values in the previous layers for the given atom or pair, a new A or P layer can be constructed. Note, since “f” keeps input and output within the same atom or pair, this operation preserves invariant 1 and 2, order and permutation invariance.

The next operation defined, (P->A), creates a new atom layer “y” from a pair layer “x” and uses both “f” (not necessarily the same “f” as in (A->A) or (P->P)) and “g”. For a given atom “a” the (P->A) operation can create a new A layer “y” from the set of pairs in P layer “x” that contain “a”, ie, (“a”,”p”) and (“p”, “a”), where “p” is any atom. The “f” function is applied to the values of these pairs individually, and these results are the input to the commutative function “g” that maintains the invariants 1 and 2.

The final operation defined, (A->P), creates a new pair layer “y” from an atom layer “x”. First, a pair “(a,b)” is chosen. Once again, a “f” function (arbitrary) is used that takes two inputs, the values of the atoms “a” and “b” at atom layer “x”. Two “f” functions are input to to the “g” function, which operate on the atoms “a”, “b” and “b”,”a” respectively. The function “f” is not necessarily symmetric, so “a” and “b” must take turns being the first and second argument. Thus, the invariants are maintained.

With these four operations defined, the Weave module can be defined as taking input layers P^k and A^k and using these operations to create layers A^(k+1) and P^(k+1).