Pairwise Augmentation for Molecular Property Prediction

Mini-Review and Initial Exploration

Jackson Burns

3 March 2025

This short paper aims to do a few things:

- formalize my (limited) literature review
- describe my initial efforts into this approach (especially related to `polaris`)
- briefly mention some of ideas for future directions
- encourage others to try this framework (especially for the `polaris` competition!)

Much of this paper has been adapted from the corresponding presentation in `meta/presentation` - consider scrolling through those slides first to get an idea of what Neural Pairwise Regression is.

To summarize, I'll say that Pairwise Augmentation is an emerging regression technique with significant potential for small-dataset machine learning (i.e. nearly *all* of chemistry). At a high level, it recasts the canonical regression problem $y = f(x; \theta)$ into the difference-prediction problem $y_1 - y_2 = \Delta_{1,2} = f(x_1, x_2; \theta)$, offering more training data and a (possibly) easier task for improved extrapolation. ML theorists are developing guidelines for using it while practitioners in the chemistry space are gathering empirical evidence of its efficacy. My contribution is to explore the use of neural networks in this context and provide a high-quality Python library in doing so. I've also run some initial benchmarks using the `polaris` library, and the results are promising.

Background

Surrogate models are arbitrary mathematical models that replace 'expensive' alternatives. In chemistry-world this could mean a solubility prediction model that circumvents the need to synthesize a drug candidate and actually run an experiment to determine its solubility. This 'canonical' problem is therefore a search for some function f with parameters θ which maps a vector representation of your input x into your target property y, i.e. $y = f(x; \theta)$.

For the scope of this study I'm going to focus on regression problems where y is continuous and (hopefully) approximately normally distributed. There are an uncountable number of techniques, frameworks, and methods that could constitute f, but I'll just mention a few here[1]:

- Statistical Modeling (SM) such as (regularized) linear regression
- Machine Learning (ML) such Decision Trees, Neighbors-Based, and *Ensembles*
- Artificial Intelligence (AI) such Neural Networks (NNs)

Broadly speaking, as one moves down that list one requires more data to fit an accurate model (θ increases) and results are less interpretable but one can fit 'harder' functions at better accuracy. Science domains are therefore in a tough spot - our complex, highly non-linear problems would be perfect for these methods, but our datasets are often small and cover only a small subset of feature space. We can counteract these challenges with pairwise augmentation.

[1]This terminology is *very* loose, some use these interchangeably i.e. linear regression = AI

Implementing pairwise augmentation starts by reformulating our canonical regression problem. Using all of the same techniques from above, we will now take in two x's simultaneously and predict the *difference* in their target property value, i.e. $y_1 - y_2 = \Delta_{1,2} = f(x_1, x_2; \theta)$. This reformulation allows us to then generate all pairwise combinations of our training data, increasing its amount by a power of 2. During inference, we predict the property difference of an unknown x to the known x's in training to undo the reformulation, with the additional benefit of providing a well-calibrated prediction of the uncertainty [1].

This technique has simultaneously emerged in a few different places across science in the past few years with different thoughts on how it should be implemented:

- [2]: Passed two inputs *and* the difference between the inputs $x_1 \oplus x_2 \oplus (x_1 - x_2)$, used ML
- [3]: Provided theoretical evidence that difference learning could make extrapolation easier by converting it from an out of support problem to an out of combination problem - also suggested including an operation within the model to enforce physics constraints.
- [1]: First study to formalize pairwise regression as a subfield, demonstrated use of NNs as f.
  - Same authors later simplified their proposed architecture [4]

Their are many open questions[2] about how to augment training, which physics constraints should be enforced, and how inference should be run(see [5]). Perhaps the most *impactful* decision, though, is the choice of f. For the scope of this work I'm focusing on NNs - they *theoretically* can capture all of the performance any other model could and I happen to be more familiar with them than the others.

Initial Investigation

I've built up a small Python library `nepare` that implements pairwise augmentation strategies and provides a basic NN class that can be used to actually run Neural Pairwise Regression. The code itself is quite short (this idea is not, ultimately, that complicated) and is currently focused more on flexibility so that the above questions can be explored.

Most of my initial work in benchmarking is saved and visible as Jupyter notebooks. There are a number of demonstrations in the repository (see `notebooks`) that show how to fit:

- `demo`: arbitrary 2D surfaces
- `polaris_adme`: ADME properties using a fixed molecular representation
- `polaris_chemprop_nepare`: ADME properties using a learned molecular representation with ChemProp
- `polaris_asap`: the ASAP discovery challenge with polaris

Each notebook contains extensive inline commentary about the finer points of implementing pairwise regression, but I'll highlight just two things:

- the results of `polaris_adme` and `polaris_chemprop_nepare` are ranked *quite* high on the RPPB leaderboard, only narrowly losing out to a massive foundation model
- the `polaris_asap` notebook shows everything that is needed to reproduce my submission for the ASAP Discovery x OpenADMET Competition's antiviral-potency and antiviral-admet tasks. The exact commits showing the run notebook are here for the former and here for the latter.

Future Directions

Obviously a random spattering of decent results aren't really enough to demonstrate the efficacy of this approach. First and foremost will be running more and more diverse benchmarks to try and understand the failure modes of this approach.

In addition to the many questions mentioned in the background, there are a host of possible approaches to enforcing loop consistency in the trained model. This refers to the fact that any closed loop of predicted

[2]The companion slides go into these in greater detail and with some helpful visualizations.

differences *should* sum to zero ($\Delta_{1,2} + \Delta_{2,n} + ... + \Delta_{n,1} = 0$). Whether or not this is possible, or even useful, has a large impact on the way that we would run inference using this framework.

Uncertainty calibration is another important aspect of this approach. The ensemble of predictions gives an intuitively well-calibrated uncertainty estimate, but this of course needs to be quantified.

Please reach out to me (GitHub, jwburns|@|mit.edu, or find my socials on my personal site) if you are interested in collaborating on any aspect of this!

Cited Works

1. Wetzel SJ, Ryczko K, Melko RG, Tamblyn I (2020) Twin neural network regression. CoRR abs/2012.14873:
2. Tynes M, Gao W, Burrill DJ, et al (2021) Pairwise difference regression: A machine learning meta-algorithm for improved prediction and uncertainty quantification in chemical search. Journal of Chemical Information and Modeling 61:3846–3857. https://doi.org/10.1021/acs.jcim.1c00670
3. Netanyahu A, Gupta A, Simchowitz M, et al (2023) Learning to extrapolate: A transductive approach
4. Wetzel SJ, Melko RG, Tamblyn I (2021) Twin neural network regression is a semi-supervised regression algorithm. CoRR abs/2106.06124:
5. Belaid MK, Wibiral T, Rabus M, Hüllermeier E (2024) Weighted pairwise difference learning