Electronic Supporting Information

Synergistic Modeling of Liquid Properties: Integrating Neural Network-Derived Molecular Features with Modified Kernel Models

Hyuntae Lim and YounJoon Jung*

Department of Chemistry, Seoul National University, Seoul 08826, Korea

March 20, 2024

1 IDAC Data Acquisition

Simplified Molecular Input Line Entry System (SMILES) strings of 2,167 organic compounds were gathered from the compound list of the DIPPR 801 database. The RDKit Python software was used to generate the raw 3D structure from the SMILES string of the given molecule, and temporal molecular mechanics (MM) optimization was performed using the Merck Molecular Force Field 94 (MMFF94)[1]. Extensive density functional theory (DFT) calculations using Gaussian 16 software were performed on the MM-optimized 3D structures at the BP86/def2-TZVP level. We utilized an open-source implementation of the COSMO-SAC model[2] to generate surface charge moments from the DFT calculation results and evaluate the infinite dilution activity coefficient (γ^{∞}) of each solvent-solute pair. 4,695,889 IDAC data points of IDAC were collected from the COSMO-SAC calculations.

2 Machine Learning Details

For the pretrained feature generation, we utilized the D-MPNN model that is implemented in the chemProp 1.5.1 Python package[3, 4] to extract the 128-dimensional vector

*yjjung@snu.ac.kr

Hyperparameter	Value
MPNN dimension	128
MPNN depth	3
FFN dimension	256
FFN depth	2
activation function	tanh
aggregation	sum
dropout	0.1
batch size	128
epochs	30

Table 1: List of pretraining hyperparameters.

representation from the given chemical structure. We allowed the two MPNN layers for the solvent and solute molecules to share their trainable parameters, enabling both layers to yield identical molecular features. After the FES/IDAC pretraining task, the output value of the last D-MPNN layer was extracted and reused for the fine-tuning model. We did not perform extensive hyperparameter optimization during pretraining. Conversely, we chose an arbitrary set of hyperparameters that provided satisfactory FES predictions with an RMSE of ~ 0.5 kcal/mol. For the fine-tuning stage, we used the LightGBM[5] software to build the gradient boosting model, and the JAX Python package[6] to implement the kernel regression model. A thorough hyperparameter search and optimization task was performed for the two fine-tuning models using the Bayesian optimization algorithm[7]. We also used the Scikit-learn[8] software as the machine learning front end, particularly for CV sampling and error evaluations.

References

[1] Landrum, G. RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling.

[2] Bell, I. H.; Mickoleit, E.; Hsieh, C.-M.; Lin, S.-T.; Vrabec, J.; Breitkopf, C.; Jäger, A. A Benchmark Open-Source Implementation of COSMO-SAC. *Journal of Chemical Theory and Computation* **2020**, *16*, 2635–2646.

[3] Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M.; Palmer, A.; Settels, V.; Jaakkola, T.; Jensen, K.;

Barzilay, R. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling* **2019**, *59*, 3370–3388.

[4] Stocker, S.; Csányi, G.; Reuter, K.; Margraf, J. T. Machine learning in chemical reaction space. *Nature Communications* **2020**, *11*, 5505.

[5] Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.-Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems. 2017.

[6] Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; Zhang, Q. JAX: composable transformations of Python+NumPy programs. 2018.

[7] Bergstra, J.; Yamins, D.; Cox, D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proceedings of the 30th International Conference on Machine Learning. 2013; pp 115–123.

[8] Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.