DeepChEmbed: Domain Aware High Dimensional Data Clustering for Molecular Property Classification

Hang Hu [1], Yang Liu [2], Yueyang Chen [3], Robert Rallo [4]

[1] Molecular Engineering & Sciences Institute, University of Washington, Seattle, WA 98195
[2] Department of Chemistry, University of Washington, Seattle, WA 98195
[3] Department of Electrical & Computer Engineering, University of Washington, Seattle, WA 98105
[4] Advanced Computing, Mathematics, and Data Division, Pacific Northwest National Laboratory, Richland, WA 99352

Overview

DeepChEmbed is an open-source python package which develops new types of chemical embeddings for the purpose of improving the classification of chemical properties, such as biodegradability, toxicity and etc.

GitHub: https://github.com/chembed/DeepChEmbed

Motivations

Clustering high-dimensional data has always been a challenging task; for chemists and materials scientists, properties such as biodegradability and toxicity provide useful information for novel materials design. To develop better machine learning algorithms for clustering of molecular properties, we combined the simultaneous training of a deep autoencoder with a clustering/classifying layer to produce novel chemical embeddings, which are *cluster-aware*.

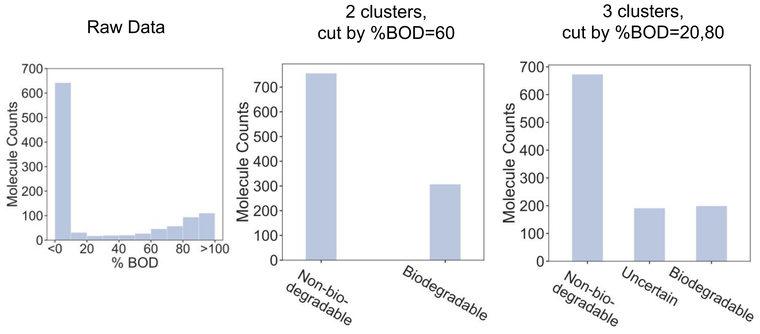
Use Cases

***With DeepChEmbed*, users can...**

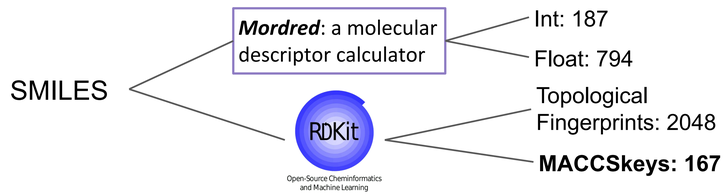
- Better distinguish between biodegradable and non-biodegradable chemicals
- Toxicity screening in pharmaceutical industry
- Visualize how well "separated" the desired property is through our model

Data Pre-processing

Dataset and cluster definition: SMILES of 1063 molecules and their % BOD (Biodegradability) from experimentalists.

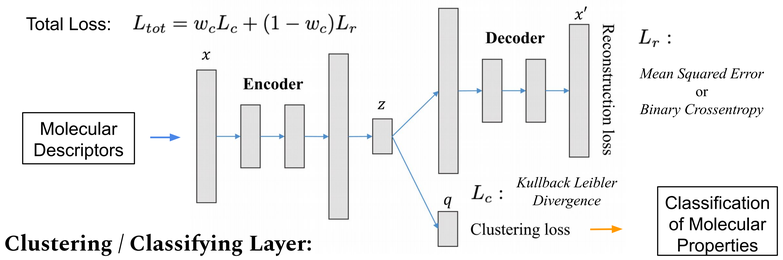


Descriptor Generation



Model Architectures

DCE Model:



Clustering / Classifying Layer:

Loss: $L_c = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$

Soft label assigning point z_i to cluster μ_j: $q_{ij} = \frac{(1 + ||z_i - \mu_j||^2)^{-\frac{1+\alpha}{2}}}{\sum_j (1 + ||z_i - \mu_j||^2)^{-\frac{1+\alpha}{2}}}$

Label hardening for clustering layer: $p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j q_{ij}^2 / \sum_i q_{ij}}$

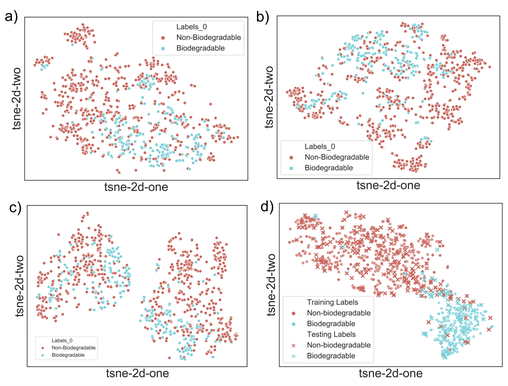
Using true label for classifying layer: $p_{ij} = \begin{cases} 1, & j = \text{true label} \\ 0, & \text{otherwise} \end{cases}$

Results & Discussion

Overview:

The combined training of autoencoder and clustering or classifying layer can impose the cluster-aware data structures. However, a significant improve on performance could only be achieved when using the true labels for this dataset.

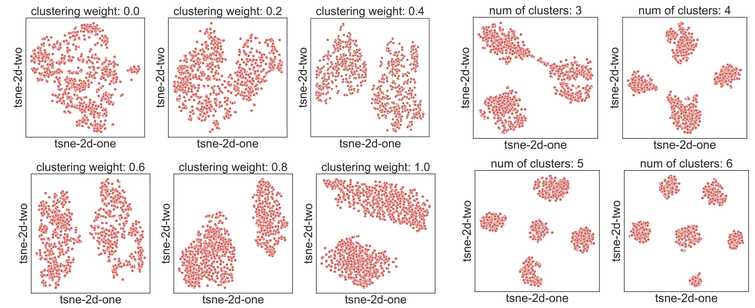
The 2-dimensional T-SNE Projections of:
a) Raw data
b) Hidden layer from Autoencoder
c) Hidden layer from combined training: Autoencoder + Clustering (w_c=0.4)
d) Hidden layer from combined training: Autoencoder + Classifying (w_c=0.4)



Results & Discussion (continued)

Unsupervised DCE: imposing the cluster-aware data structure

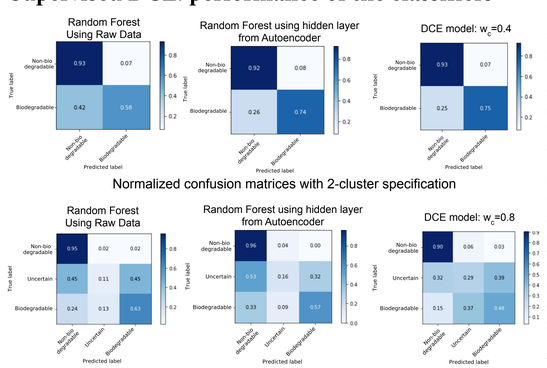
The DCE model is very effective to impose a cluster-aware data structure, irregarding to numbers of the clusters. However, the clustering weight must be tuned and balanced between obtaining a desired data structure and preserving maximum information from raw data.



The 2-dimensional T-SNE projections of hidden layers, number of clusters = 2

The 2-dimensional T-SNE projections of hidden layers, clustering weights = 0.6

Supervised DCE: performance of the classifiers



Normalized confusion matrices with 2-cluster specification

Normalized confusion matrices with 3-cluster specification

- A **75-25 train-test split** was implemented in the supervised learning.
- Normalized Confusion Matrix (predicted vs. ground true label) of the **test data** was used to evaluate the performance of each classifier.
- Random forest classifier was built on raw data (**left**) and dimension-reduced data (**middle**) with optimized parameters, and was compared with the DCE model (**right**).
- The 2-cluster cut-off (**top**) has better prediction accuracy compared to the 3-cluster cut-off (**bottom**).
- In general, the performance was improved with DCE model.

Future Work

- Coupling with other types of autoencoders, such as convolutional autoencoder, *etc.*
- Coupling with other classification algorithms, such as support vector machines, *etc.*
- Developing "interpretable" embeddings: cooperated with the chemical meanings
- Using a larger datasets with/without true labels

References

Dependencies: Pandas, Numpy, Matplotlib, Seaborn, RDKit, Modred, Seaborn, Keras, TensorFlow, Sklearn (all open-source).

Publication: Xifeng Guo, Long Gao, Xinwang Liu, Jianping Yin. Improved Deep Embedded Clustering with Local Structure Preservation. IJCAI 2017. https://github.com/XifengGuo/IDEC

DeepChEmbed would like the thank UW DIRECT and the Clean Energy Institute at the University of Washington.