To use Morgan Fingerprint to convert the smiles to numerical vector representations that can be used as input for an SVM model in Python, you can use the RDKit library to calculate the Morgan Fingerprint for each molecule in the dataset. Here is an example of how to do this:

from rdkit import Chem

from rdkit.Chem import AllChem

# Create an empty list to store the fingerprint vectors

fingerprints = []

# Loop through each molecule in the dataset

for smiles in dataset:

# Convert the smiles to a RDKit molecule

mol = Chem.MolFromSmiles(smiles)

# Calculate the Morgan Fingerprint for the molecule

fp = AllChem.GetMorganFingerprint(mol, 2)

# Add the fingerprint vector to the list

fingerprints.append(fp)

# Print the list of fingerprint vectors

print(fingerprints)

If I already get a numerical vector representation of the Morgan Fingerprint. Then how to use that as input to support vector machine to do prediction in Python?

from sklearn import svm

# Create the SVM model

model = svm.SVC()

# Train the model using the Morgan Fingerprint vectors as input

# and the corresponding labels as output

model.fit(fingerprints, labels)

# Use the trained model to make predictions on new data

predictions = model.predict(new\_fingerprints)

To use a numerical vector representation of a Morgan fingerprint as input to a support vector machine (SVM) model for prediction in Python, you will need to follow these steps:

1. Split your data into a training set and a test set. You will use the training set to train the SVM model, and the test set to evaluate the model's performance. You can use functions like train\_test\_split from the sklearn.model\_selection module to split your data.

2. Create an SVM model object and specify the hyperparameters for the model. You can use the SVC class from the sklearn.svm module to create an SVM model. You will need to specify the kernel type, the regularization parameter, and any other hyperparameters you want to use.

3. Train the SVM model on the training set. You can do this using the fit method of the SVM model object, passing in the training data and labels as arguments.

4. Use the trained SVM model to make predictions on the test set. You can do this using the predict method of the SVM model object, passing in the test data as an argument.

5. Evaluate the performance of the model on the test set. You can use functions like accuracy\_score from the sklearn.metrics module to evaluate the model's accuracy, precision, recall, and other performance metrics.

Regarding labels, in machine learning, a label is a categorical or continuous value that is associated with a data sample. In the context of an SVM model, the labels are used to train the model to classify the data into different categories. For example, if you are building an SVM model to predict whether a molecule is active or inactive against a particular target, the labels could be "active" and "inactive". You will need to provide the labels for your data when training the SVM model.

To get the labels for your data, you will need to have a way of assigning labels to your data samples. This will depend on the specific problem you are trying to solve and the data you have available. For example, you might have a separate dataset that contains the labels for your molecules, or you might need to manually annotate the data yourself. Once you have the labels, you can use them to train and evaluate the SVM model.

import numpy as np

from sklearn.multioutput import MultiOutputRegressor

from sklearn.model\_selection import train\_test\_split, RandomizedSearchCV

# Load the data

X = np.load("fingerprint\_data.npy")

y = np.load("task\_labels.npy")

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

# Create the multi-task learning model

model = MultiOutputRegressor()

# Use RandomizedSearchCV to tune the hyperparameters

param\_distributions = {'n\_estimators': [10, 50, 100, 200],

'max\_depth': [None, 2, 4, 6, 8]}

search = RandomizedSearchCV(model, param\_distributions, n\_iter=10, cv=5)

# Fit the model to the training data

search.fit(X\_train, y\_train)

# Make predictions on the testing data

y\_pred = search.predict(X\_test)

This code creates a multi-task learning model using the MultiOutputRegressor class, and tunes the model's hyperparameters using RandomizedSearchCV. The model is then fit to the training data and used to make predictions on the testing data. The specific parameters used for RandomizedSearchCV and the multi-task learning model can be adjusted as needed for your particular use case.

In the context of using RandomizedSearchCV to tune the hyperparameters of a machine learning model, best\_estimator\_ is an attribute of the RandomizedSearchCV object that returns the best estimator (model) found during the search.

1. 程序跑的非常慢，得到一个model的结果可能需要20到30分钟。所以测试加改进要花很多时间

经过使用google colab,确实快多了，然后现在还在调试hyper-parameter

1. 改进Data process,

我目前是发现missing value只有几十个，6000多个数据，我打算直接把这些值给删除，然后我还打算写个代码把outlier，就是在这个数据中值偏大或者偏小，异常的值给删除。

1. 增加几个不同的model做比较，比如Random Forest, XGBoost。

每个model的优缺点不一样，我之后再研究研究别的paper用的，然后在进行修改，目前我已经训练了5个model了。Multi task learning有些麻烦，我还在弄！

New plan: create a Correlation coefficient heatmap.

* Support Vector Machines (SVMs) are a type of linear classifier that find the best boundary between classes in a high-dimensional feature space. SVMs are often used for binary classification problems, but can also be extended to multi-class problems.
* Gaussian Process Regression (GPR) is a type of non-parametric regression model that models the relationship between the input features and the output variable as a Gaussian process. GPR is often used for regression problems where the relationship between the input features and the output variable is non-linear.
* Multi-task Learning with Neural Networks is a type of neural network architecture that is designed to handle multiple tasks simultaneously. The neural network shares some of its layers between tasks, which allows it to leverage the relationships between tasks to improve its overall performance. Multi-task learning is often used when multiple related tasks can be solved with the same input data, such as when making predictions for different properties of a chemical compound.
* Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to make a final prediction. Random Forest is often used for classification and regression problems and is known for its ability to handle large amounts of data and its robustness to noisy or irrelevant features.
* XGBoost (Extreme Gradient Boosting) is an implementation of gradient boosting, which is an ensemble learning method that builds multiple decision trees and combines their predictions to make a final prediction. XGBoost is often used for classification and regression problems and is known for its fast training times and high prediction accuracy.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

* 1. 昨天谈说到参考的paper使用的MTL，因为一开始我以为我不能用他们的code，得全部自己写，所以我就当时就直接没有在关注他的code，我只是用了这个paper得到的结论，然后想通过更多不同的机器学习的模型去验证那个结果。因为我的数据量肯定没他那个paper那么大，所以很多地方都得不一样。比如model的设计，超参数的设计。
  2. Paper的MTL使用了一个可变层数的神经网络，层数的范围是1到2之间进行随机选择，每一层由一个Dense层、一个ReLU激活函数和一个Dropout层组成，最后再加上一个输出层。根据超参数搜索的结果，实际的神经网络层数和每层的神经元个数是不固定的。

使用了一个固定的3层神经网络，分别包括一个64个神经元的输入层、一个32个神经元的隐藏层和一个输出层。这种方式相对于上面的可变层数的方式，不需要进行超参数搜索，更容易训练和调试。

固定的神经网络具有更少的超参数需要调整，训练和调试也更加容易。

数据集不是非常大的时候用固定的神经网络更好！

不过我现在也不确定我的数据量，应该用哪个更好，我也可以使用可变层数的神经网络试试，最后看训练出来的结果是否有固定的好。

* 1. PPT里面，如果讨论到一些结果性的东西，需要进行相应的citation。

比如昨天hanfeng说的，在我说RF和 XGB他们的计算成本很高，需要引用吗？不可以直接说对吗?

* 1. 答辩的时间。
  2. 谢谢JJ的推荐信。我目前拿到了cmu的计算机软件工程硅谷项目的offer。

Based on the paper "Polymer informatics with multi-task learning", I plan to do research by myself. I also get part of data from this paper. The structure of data seems like below and it is in cvs. I just put part of data here, actually my whole dataset has 8 kinds of properties.

counts smiles property value 0 [\*]CC([\*])C Eat -5.14 1 [\*]CC([\*])F Eat -5.18 2 [\*]CC([\*])(F)F Eat -5.21 3 [\*]C(F)C([\*])(F)F Eat -5.11 ... 644 [\*]Oc1ccc(C(C)(C)c2ccc(OC([\*])=O)cc2)cc1 Xc 16.83 645 [\*]Oc1ccc(C(C)=NN=C(C)c2ccc(OC(=O)OCCCCCCOC([\*])=O)cc2)cc1 Xc 44.64 646 [\*]Oc1ccc(C(C)=NN=C(C)c2ccc(OC(=O)OCCCCCCCCOC([\*])=O)cc2)cc1 Xc 34.42 647 [\*]Oc1ccc(C(C)=NN=C(C)c2ccc(OC(=O)OCCCCCCCCCCOC([\*])=O)cc2)cc1 Xc 28.99 ... 1001 [\*]C=C(F)C(F)(F)C[\*] Egc 5.7867 1002 [\*]C(F)(F)C1(F)C(F)(F)C([\*])(F)C(F)(F)C(F)(Cl)C1(F)F Egc 7.2365 1003 [\*]C(F)(F)C1(F)C(F)(F)C([\*])(F)C1(F)F Egc 7.1313 1004 [\*]C(F)(F)C1(F)C(F)(F)C([\*])(F)C(F)(F)C1(F)F Egc 7.2142 ...

The paper use QSPR fingerprints to make those data figerprinted, then use machine learning to do prediction. Now I want to use Morgan fingerprints to do this. Can you help me use Morgan fingerprints to do this and explain each step for me, and also provide all the code in Python of jupyter notebook ?

def smiles\_to\_morgan(smiles, radius=2, nBits=2048): mol = Chem.MolFromSmiles(smiles) fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits) # print(fp) return fp data['morgan'] = data['smiles'].apply(smiles\_to\_morgan) after finishing the Morgan fingerprints like the code above, based on the paper Based on the paper "Polymer informatics with multi-task learning", can you help me built Support vector Regression single task (SVR-ST) to predict each properties of those 8 properties. And also use RandomizedSearchCV to optimize the hyperparameter. Then draw a figure that show the RMSE for each properties. to avoid error, could use this code when Loop through each property and train a SVR-ST model : property\_data = data[data['property'] == property\_name]. choose Crossvalidation CV = 5; And 8 properties is : property\_names = ['Eat', 'Xc', 'Egc', 'Egb', 'Eea', 'Ei', 'nc','eps']

def smiles\_to\_morgan(smiles, radius=2, nBits=2048): mol = Chem.MolFromSmiles(smiles) fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius, nBits=nBits) # print(fp) return fp data['morgan'] = data['smiles'].apply(smiles\_to\_morgan) after finishing the Morgan fingerprints like the code above, based on the paper Based on the paper "Polymer informatics with multi-task learning", can you help me built Neural Network with Multi-task learnning(NN-MT2) to predict those 8 properties. the NN-MT2 model uses a concatenation-based MT architecture that takes in the fingerprint vector and a selector vector S, outputting only the selected polymer property. The selector vector has 8 components where one component is 1 and the rest 0. NN models has two dense layers, followed by a parameterized ReLU activation function and a dropout layer with rate 0.5. The Hyperband method optimized the two dense layers to 224 and 160 neurons for NN-MT2 model. An additional dense layer was added with 8 for the NN-MT2 model to resize the output layer. And also use RandomizedSearchCV to optimize the hyperparameter. Then draw a figure that show the RMSE for each properties. to avoid error, could use this code when Loop through each property and train the model : property\_data = data[data['property'] == property\_name]. choose Crossvalidation CV = 5; And 8 properties is : property\_names = ['Eat', 'Xc', 'Egc', 'Egb', 'Eea', 'Ei', 'nc','eps']

1. 那我目前paper主要的研究目的就是通过实现两种不同的fingerprint（Morgan和PG）在我build的5种机器学习上做预测，然后对结果进行比较和分析，不仅可以验证paper的结果，还能进一步判断不同的fingerprint对ML预测的精确度的影响！

why each ML algorithm works differently and have different accuracy, is it due to the fingerprints or is it due to the classification algorithm?

手机屏幕截图

描述已自动生成

文本

描述已自动生成