





TASK 4

Transfer learning

You are in the group  Importer consisting of  bosliu (bosliu@student.ethz.ch (mailto://bosliu@student.ethz.ch)),  hanwan (hanwan@student.ethz.ch (mailto://hanwan@student.ethz.ch)) and  junzhe (junzhe@student.ethz.ch (mailto://junzhe@student.ethz.ch)).

1. READ THE TASK DESCRIPTION

2. SUBMIT SOLUTIONS

3. HAND IN FINAL SOLUTION

1. TASK DESCRIPTION

DISCOVERING FUTURE SOLAR ENERGY MATERIALS

Welcome to task #4, the last project of this year's Introduction to Machine Learning course at ETH!

Solar power is one of the most promising renewable energy sources for reducing our dependence on fossil fuels. Unfortunately, most modern solar cells are built on *silicon*. These materials are expensive, rigid, and difficult to produce. On the other hand, *carbon*-based solar cells could be cheap to produce, flexible, transparent and easy to manufacture. There is just one problem: no *known* organic (i.e. carbon-based) photovoltaic molecules are as efficient as their silicon counterparts. The search for promising material candidates is more relevant than ever.

What makes a given material (i.e. a molecule) useful for solar energy depends largely on one quantity that measures the efficiency of the molecule. The so-called **HOMO-LUMO gap** denotes the difference in energy between the highest occupied molecular orbital (HOMO) and the lowest unoccupied molecular orbital (LUMO). The larger the HOMO-LUMO gap, the more efficient the material for downstream use in solar cells. A data set of values mapping the SMILES (https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system) string of a molecule to its HOMO-LUMO gap may look like this:

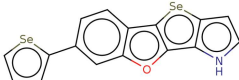
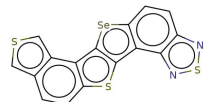
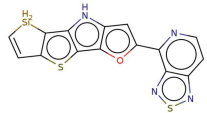
SMILES string of molecule	Visualization	E _{LUMO}	E _{HOMO}	HOMO-LUMO gap
<chem>c1c[se]c(c1)-c1ccc2c(c1)oc1c3[nH]ccc3[se]c21</chem>		-2.3036	-5.8230	3.5194
<chem>c1scc2c1ccc1sc3c([se]c4ccc5nsnc5c34)c21</chem>		-3.2578	-4.9441	1.6863
<chem>[SiH2]1C=Cc2sc3c([nH]c4cc(oc34)-c3nccc4nsnc34)c12</chem>		-3.1360	-5.0076	1.8716
...

Table 1: Dataset of molecules and their HOMO and LUMO energies. Images generated by: <http://cdb.ics.uci.edu/cgi-bin/Smi2DepictWeb.py>

TASK: TRANSFER LEARNING

In this project, your task is to discover promising materials for organic solar cells by *predicting* the HOMO-LUMO gap from a molecule description. Traditionally, density functional theory is used to estimate properties of molecules. However, this is computationally very expensive. Instead, we would like to use machine learning to predict the HOMO-LUMO gap from the molecule description directly. Using the machine learning model you will develop, we can efficiently sift through millions of potential molecule configurations and select the most promising ones for downstream experiments.

There is one catch: we are only given a very limited number of 100 labeled data points, i.e., 100 (molecule, **HOMO-LUMO gap**) tuples. However, there is an *additional* data set available that you can be used to improve our accuracy of predicting the HOMO-LUMO gap. Specifically, we have another related data set of 50,000 labeled (molecule, **E_{LUMO}**) pairs to supplement the 100 data points for solving our objective. Note that in the second, larger data set, the labels (i.e. the “y-values”) are *not* the quantity we are interested in (i.e., not the HOMO-LUMO gap), but instead another, highly related quantity: the LUMO energy level E_{LUMO} alone (cf. Table 1). The molecules in both data sets are not overlapping.

The key insight is that the ability of predicting the LUMO energy given a molecule description can *transfer* to the ability of predicting the HOMO-LUMO gap. In particular, molecule *features* that are predictive of the LUMO energy are likely also predictive of the HOMO energy and therefore also the HOMO-LUMO gap. In other words, being given a related data set, we should not start “from scratch” when solving the prediction task of the HOMO-LUMO gap from only 100 examples. Instead, the features learned on one task should transfer to the other task, and thus the larger data set of (molecule, E_{LUMO}) pairs should be used to improve prediction of the HOMO-LUMO gap. In machine learning parlance, this is called *transfer learning*. This setting is very realistic in many practical applications, where we rarely have a lot of data for the exact task we are interested in.

In class, you have learned about unsupervised representation learning, for instance, using an autoencoder. Similar ideas may help you to extract useful feature representations for molecules from the large dataset to improve the accuracy of predicting the HOMO-LUMO gap. The difference here is that you should “transfer” your knowledge to using the additional label information in the large data set, too ;-). For instance, analogous to the encoder of an autoencoder, you can re-use the feature representation of the last layer when training a neural network model to predict the LUMO energy. This is common practice in, for example, computer vision. Given the difference in sizes of the two available data sets, your performance on the test set likely depends much more on how you make use of the large, pretraining data set versus the small, training data set. We wrote a small handout for you that explains this solution idea further.

[Handout \(/static/task4_hint_on_pretraining.pdf\)](/static/task4_hint_on_pretraining.pdf)

DATA FORMAT

As mentioned, we provide two training data sets. A small data set with 100 rows called `train.csv` containing HOMO-LUMO gap values for 100 molecules. And a large data set with 50,000 rows called `pretrain.csv` containing LUMO energy levels for 50,000 molecules. In addition, there is a third file called `test.csv` containing 10,000 unseen molecule descriptions without labels. Your task is to provide predictions of the HOMO-LUMO gap for the molecules given in `test.csv`. You should submit a `csv` file containing the predicted HOMO-LUMO gap in the same order as in `test.csv`. The `csv` file should contain only one floating point value in each separate line. For your convenience, we provide a sample submission file with random values.

To make employing a machine learning model for this task easier, each row of the three data sets not only contains the SMILES string of the molecule (and a target value in `pretrain.csv` and `train.csv`) but also 1,000 numerical features of the given molecule. The columns are thus:

```
smiles, feature_0000, feature_0001, ..., feature_0998, feature_0999, target
```

where the `target` columns is `lumo_energy` for the `pretrain` set and `homo_lumo_gap` for the `train` set. For the `test` set, the `target` columns is, of course, missing. The features were created with basic chemical fingerprints using the RDKit package (<https://github.com/rdkit>). To further boost your performance, you can try to engineer more predictive features using the package. However, passing the project with full credit is possible using only the features we provide.

You can download the project material here:

[Download \(/static/task4_hr35z9.zip\)](/static/task4_hr35z9.zip) (4.4 MB)

In summary, the folder will contain the following files

- **pretrain.csv**: pretraining set with 50,000 molecules
- **train.csv**: training set with 100 molecules
- **test.csv**: test set with 10,000 molecules
- **sample.csv**: a sample submission file in the correct format

EVALUATION

We are interested in accurate prediction of the HOMO-LUMO energy gap value. Your submitted predictions will be evaluated by the *root mean square error (RMSE)* to the true HOMO-LUMO gap value computed the density functional theory. Lower is better. The RMSE is defined as

$$\text{RMSE}(\mathbf{y}^*, \mathbf{y}) = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i^* - y)^2}$$

To calculate the RMSE score, we use the scikit-learn implementation:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_true, y_predicted, squared=False)
```

GRADING

We provide you with **one test set** for which you have to compute predictions. We have partitioned this test set into two parts (of the same size) and use it to compute a *public* and a *private* score for each submission. You only receive feedback about your performance on the public part in the form of the public score, while the private leaderboard remains secret. The purpose of this division is to prevent overfitting to the public score. Your model should generalize well to the private part of the test set. When handing in the task, you need to select which of your submissions will get graded and provide a short description of your approach. This has to be done **individually by each member** of the team. We will then compare your selected submission to our baseline. This project task is graded with either **pass (6.0)**, **partial pass (4.0)** or **fail (2.0)**. To fully pass the project (grade: 6.0), you need to perform better than the baseline in both private and public score. If you only outperform the baseline in either the private or the public score, you will get a partial pass (grade: 4.0). In addition, for the pass/fail decision, we consider the code and the description of your solution that you submitted. The following **non-binding** guidance provides you with an idea on what is expected to pass the project: If you hand in a properly-written description, your source code is runnable and reproduces your predictions, and your submission performs better than the baseline, you can expect to have passed the assignment.

⚠ Make sure that you properly hand in the task, otherwise you may obtain zero points for this task.

FREQUENTLY ASKED QUESTIONS

🕒 WHICH PROGRAMMING LANGUAGE AM I SUPPOSED TO USE? WHAT TOOLS AM I ALLOWED TO USE?

You are free to choose any programming language and use any software library. However, **we strongly encourage you to use Python**. You can use publicly available code, but you should specify the source as a comment in your code.

🕒 AM I ALLOWED TO USE MODELS THAT WERE NOT TAUGHT IN THE CLASS?

Yes. Nevertheless, the baselines were designed to be solvable based on the material taught in the class up to the second week of each task.

🕒 IN WHAT FORMAT SHOULD I SUBMIT THE CODE?

You can submit it as a single file (main.py, etc.; you can compress multiple files into a .zip) having max. size of 1 MB. If you submit a zip, please make sure to name your main file as *main.py* (possibly with other extension corresponding to your chosen programming

language).

🕒 WILL YOU CHECK / RUN MY CODE?

We will check your code and compare it with other submissions. We also reserve the right to run your code. Please make sure that your code is runnable and your predictions are reproducible (fix the random seeds, etc.). Provide a readme if necessary (e.g., for installing additional libraries).

🕒 SHOULD I INCLUDE THE DATA IN THE SUBMISSION?

No. You can assume the data will be available under the path that you specify in your code.

🕒 CAN YOU HELP ME SOLVE THE TASK? CAN YOU GIVE ME A HINT?

As the tasks are a graded part of the class, **we cannot help you solve them**. However, feel free to ask general questions about the course material during or after the exercise sessions.

🕒 CAN YOU GIVE ME A DEADLINE EXTENSION?

⚠️ We do not grant any deadline extensions!

🕒 CAN I POST ON MOODLE AS SOON AS HAVE A QUESTION?

This is highly discouraged. Remember that collaboration with other teams is prohibited. Instead,

- Read the details of the task thoroughly.
- Review the frequently asked questions.
- If there is another team that solved the task, spend more time thinking.
- Discuss it with your team-mates.

🕒 WHEN WILL I RECEIVE THE PRIVATE SCORES? AND THE PROJECT GRADES?

We will publish the private scores, and corresponding grades before the exam the latest.