Basic Machine Learning

UPDATED on June 14th: Task 6.2

6.1 (E) Feature Extraction and Basic Machine Learning

During the exercise session we will show how to use the sliding window approach to encode a labeled protein sequence for machine learning. The slides will explain the ideas and the different steps to prepare input data for machine learning devices using scikit-learn. We will also show some simple data cleaning procedures. The other main part walks you through the steps from the input data set to a prediction and how to report prediction results. For reasons of simplicity we use a Logistic Regression learner. Further optimization like hyperparameter tuning or rebalancing of the dataset is beyond the scope of this exercise sheet. As well as more advanced methods of performance evaluation.

6.2 (H) FeatureGeneration

Your task is to complete the class FeatureGenerator and implement the missing functions as specified. For reasons of simplicity we provide already a code skeleton. Use the function `generatefasta_reader` to conveniently create an input function. Of course you are also free to implement your own one. The input data file is in the subdirectory data. It contain data about membran proteins. Each record consists of three lines, the header line starting with >, the protein sequence and the line with the sequence labels describing the state. There are the following state labels: 1: non-membrane, inside the cell (cytoplasmic), 2: non-membrane, outside the cell, 0: non-membrane, unknown topology, H: transmembrane helix, L: membrane, re-entrant loop, U: unknown (unresolved in PDB), h: considered border between to transmembrane helices.

To test your code we chose an object-oriented approach if this looks artificial. Most of the methods modify the inner state of the object, so that the execution order matters. We chose this way because it allows us to check for intermediate results. The places where you should put your code are either marked with #TODO or ....

(a) Implement the constructor (`__init__`) -method of the FeatureGenerator class and call `__sliding_window` to initialize the object feature table. Create a fasta_reader method to read in the data.
(b) Implement the getter method to return the feature table and carefully look at the comments in _sliding_window

(c) Implement _sliding_window as specified in the comments

(d) Complete the method create_dataframe and add the requested columns labels. See the comments.

(e) Implement the body for remove_rows_from_df

(f) Implement the body for modify_cells_from_df

(g) Implement the body for create_full_binarized_dataframe

(h) Implement the body for create_partial_binarized_dataframe

(i) Implement the body for encode_multi_class_label

(j) You can use this to create a data set with half_win_size of three, remove all residues with state ‘U’, relabel residues from ‘h’ to ‘H’ and recode the state as multi-class label. (This not mandatory since we do something similar in our tests.) Please use also the provided sample output. The sample output has been generated with the provided data file named opm.tmps_stud.fasta, so you have to adjust the main function. There was a bug in the sample output and it has been replaced with the correct values now. For reasons of readability we had to adjust some linebreaks.

6.3 (H) Machine Learning

In this task we build on the results from task 2. The input data for the Machine_Learner is a data frame with binarized attributes and a multi-class label. The constructor takes only an integer to define the random number used at various steps.

(a) Implement the __init__-method. It should store the seed for further usage. Initialize the internal classifier as a LogisticRegression object with seed als value for random_state

(b) Provide data and label information in the set_df method. df is described above

(c) Complete initialize_data to create the numpy arrays use for needed for the machine learning

(d) Implement the body of initialize_training_test_sets to generate training and test sets of features and labels, typically called X_train, X_test, y_train, y_test. Mind the seed.

(e) Train the classifier.

(f) Implement the body of make_predictions to predict on the test set

(g) Implement the body of get_confusion_matrix to return the confusion matrix.

(h) Implement the body of get_classification_report to return the classification report. Do not worry about the warning, this is due to the fact that some classes are not represented in the test fold.