Class Data - Documentation

The Data class provides a convenient way to handle biological sequence and structure data for multiple classes. Sequence and structure data are automatically converted into one-hot encoded matrices and split into training/validation/test sets. The data object can then be passed to Grid_Search or Model objects for easy training and evaluation.

Input format: Data objects accept raw strings in fasta format as input for sequence and structure data or optionally position-weight matrices for structure data (see ___init___ function). Strings can contain all uppercase alphanumeric characters and the following special characters: “()<>,.|*“. Additional handcrafted features may be added using the load_additional_data function.

Methods - Overview

name	description
___init___	Load the sequences and split the data into 70%/15%/15% training/validation/test.
train_val_test_split	Randomly split the data into training, validation and test set.

name	description
load_additional_data	Add additional numerical or categorical features to the network (for each sequence as a whole).
load_additional_positionwise_data	Add additional numerical features to the network (for each nucleotide in a sequence).

name	description
get_labels	Get the labels for a subset of the data.
get_summary	Get an overview of the training/validation/test data for each class.

___init___

```python
def __init__(self, class_files, alphabet, structure_pwm=False)
```

Load the sequences and split the data into 70%/15%/15% training/validation/test.

If the goal is to do single-label classification a list of fasta files must be provided (one file per class, the first file will correspond to 'class_0'). In this case fasta headers are ignored. If the goal is multi-label classification a single fasta file must be provided and headers must indicate class membership as a comma-separated list (e.g. header '>0,2' means that the entry belongs to class 0 and 2).

For sequence-only files fasta entries have no format restrictions. For sequence-structure files each sequence and structure must span a single line, e.g.:

>header
CCCCAUAGGGG
((((...))))

in which the second line contains the sequence and the third line the structure. **Important: All sequences in all files must have the same length.**

The provided alphabet must match the content of the fasta files. For sequence-only files a single string (e.g. 'ACGT' or 'ACGU') should be provided and for sequence-structure files a tuple should be provided (e.g. ('ACGU', '().')). Characters that are not part of the provided alphabets will be randomly replaced with an alphabet character.

We support all uppercase alphanumeric characters and the following additional characters for alphabets: "()<>,.|*". Thus, it is possible to use and combine (in the sequence-structure case) arbitrarily defined alphabets as long as the data is provided in the described fasta format. In particular, this means the usage of the package is not restricted to RNA secondary structure (this is only an example). If you have structure information for DNA or protein data that can be encoded by some alphabet, similar to RNA structure information, you can apply the package to this kind of data as well.

If you don't want to work with a single minimum free energy structure (as some RNA structure prediction tools can output multiple predictions) you can also provide a position-weight matrix representing the structure instead of a single string (matrix entries must be separated by a space or tab):

```
>header
GGGGUUCCCC
0.9 0.8 0.7 0.9 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.2 0.7 0.8 0.9
0.1 0.2 0.3 0.1 1.0 1.0 0.8 0.3 0.2 0.1
```

If you provide "()." as the alphabet the first line of the matrix given above will correspond to "(", the second to ")" and the third to ".". Each column of the matrix must add up to 1. Again, we don't restrict the usage of the package to RNA, therefore the matrix given above can represent whatever you want it to represent, as long as you provide a valid alphabet.

parameter	type	description
class_files	str or [str]	A fasta file (multi-label) or a list of fasta files (single-label).
alphabet	str or tuple(str,str)	A string for sequence-only files and a tuple for sequence-structure files.

parameter	type	description
structure_pwm	bool	Are structures provided as single strings (False) or as PWMs (True)?

train_val_test_split

```python
def train_val_test_split(self, portion_train, portion_val, seed = None)
```

Randomly split the data into training, validation and test set.

Example: setting portion_train = 0.6 and portion_val = 0.3 will set aside 60% of the data for training, 30% for validation and the remaining 10% for testing. Use the seed parameter to get reproducible splits.

parameter	type	description
portion_train	float	Portion of data that should be used for training (<1.0)
portion_val	float	Portion of data that should be used for validation (<1.0)

parameter	type	description
seed	int	Seed for the random number generator.

load_additional_data

```
def load_additional_data(self, class_files, is_categorical=False, categories=None, standardize=False)
```

Add additional numerical or categorical features to the network (for each sequence as a whole).

For every input sequence additional data can be added to the network (e.g. location, average sequence conservation, etc.). The data will be concatenated to the input of the first dense layer (i.e. additional neurons in the first dense layer will be created). Input files are text files and must contain one value per line, e.g.:

0.679
0.961
0.065
0.871
…

The number of provided files must match the fasta files provided to the __init__ function (e.g. if you provided a list of 3 files to __init__ you must provide a list of 3 files here as well) and the number of lines in each file must match the number of entries in the corresponding fasta file. If you want to add multiple features simply call this function multiple times.

Interpreting the influence of arbitrary additional data for a neural network is hard and at the moment we don't provide any means to do so. You should run your model with and without the additional data and check if the predictive performance improves. In general, if you have many handcrafted features you might want to consider using a different machine learning technique.

parameter	type	description
class_files	str or [str]	A text file (multi-label) or a list of text files (single-label).

parameter	type	description
is_categorical	bool	Is the provided data categorical or numerical?
categories	[str]	A list containing all possible categories (only needed if is_categorial == True).
standardize	bool	Should the z-score be computed for numerical data?

load_additional_positionwise_data

```
def load_additional_positionwise_data(self, class_files, identifier, standardize=False)
```

Add additional numerical features to the network (for each nucleotide in a sequence).

For every position in an input sequence additional numerical data can be added to the network (e.g. ChIP-seq signal, conservation for every nucleotide). The data will be added to the input matrix. E.g.: Using sequences of length 200 over the alphabet "ACGT" results in input matrices of size 4x200. Additional position-wise data will be added to these matrices as a new row resulting in matrices of size 5x200.

Input files are text files and must contain as many whitespace-separated values in each line as the sequences are long, e.g.:

0.679 1.223 -0.296 . . .
0.961 0.532 0.112 . . .
0.065 -0.333 -0.256 . . .
. . .

The number of provided files must match the fasta files provided to the ___init___ function (e.g. if you provided a list of 3 files to ___init___ you must provide a list of 3 files here as well) and the number of lines in each file must match the number of entries in the corresponding fasta file. If you want to add multiple features simply call this function multiple times.

Input features should be standardized in some way prior to adding them to the network, as this tends to improve the predictive performance.

In the same way network kernels are visualized as sequence motifs after the network training (based on the first 4 rows of the input matrices and using the visualize_kernel() Model function), the rows corresponding to additional features are summarized as line plots as well.

parameter	type	description
class_files	str or [str]	A text file (multi-label) or a list of text files (single-label).
identifier	str	A short feature name (will be shown in kernel output plots).

parameter	type	description
standardize	bool	Scale each column according to the interquartile range.

get_labels

```python
def get_labels(self, group)
```

Get the labels for a subset of the data.

The ‘group’ argument can have the value ‘train’, ‘val’, ‘test’ or ‘all’. The returned array has the shape (number of sequences, number of classes).

parameter	type	description
group	str	A string indicating for which subset the labels should be returned.

returns	type	description
labels	numpy.ndarray	An array filled with 0s and 1s indicating class membership.

get_summary

```
def get_summary(self)
```

Get an overview of the training/validation/test data for each class.

returns	type	description
summary	str	A tabular overview of every class.