Signal peptide predictor

This is signal peptide predictor project authored by Josip Matak and Josip Mrđen. Program was built in Python version 3.6 using different modules. Feature extraction and data processing has been done in Numpy module, while Matplotlib was useful for draw diagrams of learning process and visualizing prediction results. All machine learning techniques which mostly include working with neural networks were done using Keras, API built on top of Tensorflow library.

Installation

To install all necessary dependecies, follow next steps.

```
$ cd src
$ pip3 install requirements.txt
```

To run program, there needs to be data provided from Git, you can run it by typing.

```
$ chmod +x peptide_predictor.py
$ ./peptide_predictor.py
```

Manual

Working in peptide predictor is not so seamless as it should be, so lets run throuh some useful commands.

Next one shows help in terminal, with useful command arguments.

```
$ /peptide_predictor.py -h
```

For training peptide predictor with data in /data file, you should run next command. Window size will be default, 21, while log model are going to be stored in /models with current timestamp.

```
$ /peptide_predictor.py --train
```

To specify window length, add -w argument. Be careful because it requires odd number and may result with awkwardly large number of training samples.

```
$ /peptide_predictor.py --train -w 23
```

Specify hidden layer architecture by simply adding -a. That will result in architecture of INPUT x 128 x 64 x 32 x OUTPUT

```
$ /peptide_predictor.py --train -a 128 64 32
```

This command will start testing model stored in following folder, on default test data placed in /data/test.fa. Output will be written in “output” file, and there will be no plotting. Be sure that window size fits given model, it is default if nothing is provided.

```
$ /peptide_predictor.py --test -m model_2018-12-14-20:41:4
3 -w 23
```

If you want to change testing data, it can be done with following command. It will check file in /data + “/argument”

```
$ /peptide_predictor.py --test -m model_2018-12-14-20:41:4
3 -w 23 -f proteomes/UP000005640_9606.fasta
```

Define specific output file name with following command

```
$ /peptide_predictor.py --test -m model_2018-12-14-20:41:4
3 -w 23 -o outputfile.txt
```

Finally, you can plot each of runs in testing data with next command.

```
$ /peptide_predictor.py --test -m model_2018-12-14-20:41:4
3 -w 23 -f proteomes/UP000005640_9606.fasta --plot
```