*Supplementary information*.

Written by Peter Sarvari, MEng student, Imperial College London, UK

1. Setting up the pipeline on Windows (user-friendly)

Follow the steps below to install Python:

* 1. Use this link: <https://conda.io/miniconda.html> to download Miniconda 64-bit version with Python 3.6
  2. After installation open the Anaconda Command Prompt and type the following to install and activate rdkit:
     1. Conda create -c rdkit -n my-rdkit-env rdkit
     2. Activate my-rdkit-env
  3. Install other required libraries by typing the followings into the Anaconda Command Prompt
     1. Conda install keras
     2. Conda install biopython
     3. Pip install pydpi
     4. Conda install sickit-learn
  4. In this case, we will be using the Theano backend with Keras
     1. First, make sure Theano is installed (may have already been installed by Keras) by typing: conda install theano
     2. Second, make sure that numpy is installed by typing: conda install numpy
     3. For Theano to be used, pygpu and libgpuarray need to be downgraded from default to 0.6.2, so type:
        1. conda install pygpu=0.6.2
        2. conda install libgpuarray=0.6.2
     4. Since Keras is based on TensorFlow by default, Theano needs to be set as the backend in the \_\_init\_\_.py file in the keras\backend folder. The exact path to this folder will vary, but as an example, my path is: C:\Users\Peter\AppData\Local\conda\conda\envs\my-rdkit-env\Lib\site-packages\keras\backend. This should be done manually, by editing this \_\_init\_\_.py document, for example, by Notepad, looking for backend and specifying ‘theano’.
  5. Replace all the pydpi .py files with the ones with the corresponding name from GitHub. The pydpi files will be found in the pydpi library in site-packages. The exact path to the folder will vary, but, as an example, my path is: C:\Users\Peter\AppData\Local\conda\conda\envs\my-rdkit-env\Lib\site-packages\pydpi. This ensures that the scripts are compatible with Python 3.6.
  6. Download an editor, for example, Atom (<https://atom.io/>). Download our protein\_prediction\_pipeline.py as well as inputoutput.py, config.py, modelbuilder\_pipeline.py and featurex\_pipeline\_v3.py scripts from GitHub. Save it into a new folder, which contains your positive and negative dataset in the “.fasta” or “.text” format. Then open the python script with your editor and edit the relevant fields (outlined in the code, see point 3).
  7. Copy the path of the folder you saved the script in and put cd (standing for change directory) in front of it in the Anaconda Command Prompt. In my case, this is: cd C:\Users\Peter\Desktop\MIT\code
  8. To start executing the script, type: python protein\_prediction\_pipeline.py
  9. If you close the Anaconda Command Prompt (or restart your computer), you need to do steps b/ii) and g again!

1. Setting up the pipeline on Google Cloud (advanced)
   1. Download the Google Cloud SDK Shell from: <https://cloud.google.com/sdk/downloads>
   2. Sign into your google account, go to: <https://console.cloud.google.com> and create a project and instance as described here: <http://cs231n.github.io/gce-tutorial/>
      1. Stick to the Ubuntu system, but you can lower the specifications: 2 vCPUs, 5.75GB memory should do it
   3. Take note of your instance name
   4. Open Google Cloud SDK and type: gcloud init
   5. Follow the instructions on screen and take note of the time zone you choose
   6. Create SSH by typing something like: gcloud compute ssh --zone=us-central1-c instance-2 , where you need to modify the zone (us-central1-c) and the instance name (instance-2) to your own time zone and instance name
   7. You should see the PuTTY window popping up
   8. Install Miniconda 3 and add it to your homedirectory by typing
      1. wget [http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86\_64.sh -O ~/miniconda.sh](http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh%20-O%20~/miniconda.sh)
      2. bash ~/miniconda.sh -b -p $HOME/miniconda
      3. export PATH="$HOME/miniconda/bin:$PATH"
   9. Install required things by typing words in *italics*
      1. Install rdkit (same as under poin 1): *conda create -c rdkit -n my-rdkit-env rdkit*
      2. Activate environment: *source activate my-rdkit-env*
      3. *Conda install keras* (tensorflow should automatically come with it, luckily on linux it is compatible with rdkit!)
      4. *Conda install biopython*
      5. *Pip install pydpi*
      6. *Conda install jupyter notebook*
      7. *Conda install sickit-learn*
      8. Use the cs231n description (<http://cs231n.github.io/gce-tutorial/>) to make static IP and set firewall rule on konzol, then use nano to append to the end of the juypter init file and use paste to create the ‘\*’ (I could not figure out any other way other way to type a \* in nano other than pasting it from clipboard)
      9. *jupyter-notebook --no-browser --port=7000*
      10. Navigate to the required static IP ‘/’ <Port number> (for example, <http://164.178.50.189:7000> ) by typing it into your browser and use token given in the PuTTy command window in the hyperlink that it asks you to paste into browser (DO NOT PASTE THAT THOUGH, just use the token from the link, after ‘token=’)
   10. Upload the required files (protein\_prediction\_pipeline.py as well as inputoutput.py, config.py, modelbuilder\_pipeline.py and featurex\_pipeline\_v3.py scripts from GitHub and your positive and negative datasets) using the upload tab on the browser after opening jupyter notebook and edit the relevant fields of the Python script (see point 3)
   11. Since Python version 3 is used, we need to overwrite the print and include commands in the pydpi in the following path (your path will be the same, but the second directory will be your name rather than mine): "/home/Peter/miniconda/lib/python3.6/site-packages/pydpi " in folders drug as well as protein and “pydrug.py”. Since we have done this already, just replace the imported files in the relevant directories with the ones we uploaded to GitHub.
   12. Use CTRL-C in the PuTTY window to terminate Jupyter Notebook and type: python protein\_prediction\_pipeline.py to execute the script
   13. If you get an error (I did) while loading shared libraries, here is the solution: <https://www.cyberciti.biz/faq/debian-ubuntu-linux-wkhtmltopdf-error-while-loading-shared-libraries-libxrender-so-1/>

Basically, search for the missing libs using “apt-cache search libXrender”, which returns the missing libraries and then install them using, for example, “sudo apt-get install libfontconfig1 libxrender1”

* 1. If you close your PuTTY window (the one that appeared after you computed the ssh in step f) you need to do step f), h/iii) and i/ii) again! If you restart your computer you will also need to open Google Cloud SDK first. If you want to use Jupyter notebook you will need to do steps i/ix) and i/x) again.

1. User manual for editing the protein\_prediction\_pipeline.py script

After opening the script, look for this well outlined portion (here I have deleted the comments to make this more readable):

'''User interface'''

'''Options start'''

vector\_name = "pipeline\_default"

model\_name = "pipeline\_default"

choice = 0

reduce\_by\_similarity = 1

use\_random\_small\_sequence\_negative = 0

already\_extracted = 0

pos\_input\_name = "insert\_name"

neg\_input\_name = "insert\_name"

'''Only If prediction (choice 6) is chosen'''

predict\_pos\_input\_name = "insert\_name"

predict\_neg\_input\_name = "insert\_name"

known\_classes = 1

'''Only change if you get error regarding the sampling or if you are notified that your dataset s not balanced and asked to increase negative dataset sampling'''

sc\_1 = 2

sc\_2 = 1.25

'''Options end'''

'''User interface'''

* 1. Vector\_name refers to the name of the pickle file in which you store the extracted features of your sequences. For the positive dataset the name of the pickle file will be the vector name preceded by “pos\_” and for the negative dataset it will be the vector name preceded by “neg\_”.
  2. Model\_name refers to the names of the files that save the data associated with the trained model. This data is the model structure (json file), the weights data (h5 file), the mean and the standard deviation used for scaling the input data (preceded by “mean\_” and “std\_” respectively, pickle files). I recommend to make the vector name and model name the same, for example, if you are building a classifier with sequences having anti-parasitic effect, you could name it “pipeline\_antiparasitic”.
  3. For choice, make it equal to the number that describes the option you would like to use (see Methods section). Default is zero meaning 10-folds cross-validation.
  4. Reduce\_by\_similarity should be one, unless you have a specific negative dataset (see Methods section).
  5. Use random\_small\_sequence\_negative should be zero, unless, you would like to use the sample negative dataset provided by us. If so, you will need to download this from GitHub. This sequence is the negative dataset we used for the anti-cancer classification (see description in Data Mining section).
  6. Already\_extracted should be set to one, if you are have already run the script with these inputs (datasets) before, but now you would like to change the model (evaluation: set choice to a different number or -advanced- use a custom-written modelbuilder.py). This is to save time, since the features have already been extracted, so there is no need to do that again.
  7. Pos\_input\_name and neg\_input\_name is the name of the positive and negative fasta or txt files that contain the positive sequences (with a specific activity) and the negative sequences, respectively. For example, in the antibacterial case my neg\_input\_name is "uniprot\_3\_100\_noannotationantibacterial.fasta".
  8. Predict\_pos\_input\_name and predict\_neg\_input\_name should not be edited unless you use choice = 6 (prediction), in which case you will use one of your existing models to predict the classes of a new set of data. If you know which class the sequences in this new data belong to, put the positive sequences in a text/fasta file and make the name of the file equal to predict\_pos\_input\_name. Then put the negative sequences in another file and assign it to predict\_neg\_input\_name. If you have no such information, simply partition the sequences into two files (does not matter how you do it and they do not need to contain the same amount of sequences) and set known\_classes to zero. The probabilities you will see correspond the order of sequences in the negative dataset and then that of the positive dataset.
  9. Sc\_1 and sc\_2 and the sampling coefficients. You should only change them if you get an error message regarding the sampling or if you are asked to do so, because the classes are imbalanced. In the first case (sampling error), you should increase sc\_1 slightly (e.g. to 3 – take care than in this case the negative dataset size should be about 3 times the size of the positive dataset!) In the class imbalance case, you should increase sc\_2 (e.g. to 2). Sc\_2 should always remain smaller than sc\_1.

1. FAQ:
   1. Problem: I got out-of-bound error when the script was trying the access the elements of an array.

Solution: Make sure that your negative dataset contains at least two times as many samples as the positive one. If not, reduce sc\_1, but make sure it is bigger than sc\_2.

* 1. Question: Which featurex online code shall I use?

Answer: It depends.

* + 1. If you would like to use the 82 features, use featurex\_pipeline\_v3.py.
    2. If you would also like to use the Moreau-Broto autocorrelation, use featurex\_pipeline.py. In this case you need to download this file from GitHub too, put it in the same folder as protein\_prediction\_pipeline.py is and rewrite the import in line 13 to “import featurex\_pipeline as FX”.
    3. If you would like to get a quick baseline prediction with the amino acid composition features (20), use featurex\_pipeline\_aac.py. In this case you need to download this file from GitHub too, put it in the same folder as protein\_prediction\_pipeline.py is and rewrite the import in line 13 to “import featurex\_pipeline\_aac as FX”.
    4. Advanced: Using the featurex scripts as templates, and reading the pydpi paper, you can easily customize feature extraction.