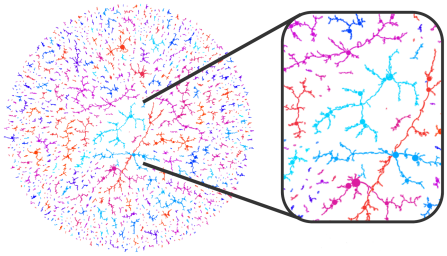
quasinet

Author: ZeD@UChicago <zed.uchicago.edu>

Description: Infer Non-local Structural Dependencies In Genomic Sequences. Genomic sequences are esentially compressed encodings of phenotypic information. This package provides a novel set of tools to extract long-range structural dependencies in genotypic data that define the phenotypic outcomes.



Introduction:

This document will walk through an example usage of the **quasinet** python package. We will start with an example set of DNA sequences. These sequences will consist of 400 basis pairs each. For each of these 400 positions/indices we will construct a conditional inference tree. We will then have a decision tree for each of these positions where the decisions are informed by the basis pairs at other positions. We will then produce some visualizations of the Qnetwork.

Requirements:

The graphviz dependency requires some software to be installed first. Please run:

```
apt-get install python-dev graphviz libgraphviz-dev pkg-config
```

The quasinet package can then be easily installed through pypi.

```
pip install quasinet
```

This should install all required python libraries necessary for this package. However, implementations of conditional inference trees are not readily available in Python. Thus, this package uses Ctrees implemented in R. We require R and two R packages to be installed: **partykit** (1.1-1) and **randomForest** (4.6-14). The version of these packages do matter. A third package, **Formula**, is necessary for partykit to work. We cannot gurantee the package will work with other versions of these R packages. To install R:

```
apt install r-base-core
```

Here are three links to three sources of required R packages.

1. https://cran.r-project.org/web/packages/Formula/index.html
2. https://cran.r-project.org/src/contrib/Archive/partykit/
3. https://cran.r-project.org/web/packages/randomForest/index.html

All three packages can be installed easily and in the same way. Download the zipped source files. Then open R and run:

```
install.packages("/path/to/source.tar.gz", repos=NULL, type="source")
```

About the example:

When the quasinet package is installed, examples will be placed in the enviroment's home directory. This documentation deals with example 2. Example 2 consists of a script and a text file containing 144 DNA sequnces. The script contains a few steps. The user can uncomment all four steps and the script will run through the entire example.

```
from quasinet import Qnet
import subprocess

responses = list(range(0,400))

#Step 1:
Qnet.fit_sequences('sequences.txt','train.csv','test.csv', test_ratio=0.2)

#Step 2:
Qnet.makeQNetwork(responses, 'train.csv','test.csv',tree_dir='tree/', VERBOSE=True)

#Step 3:
Qnet.connectQnet(responses, 0.50, '50network.dot','50network.dat',tree_dir='tree/',DEBUG=True)
Qnet.draw_Qnet('50network.dot',out_name = '50net.png')

#Step 4:
Qnet.connectQnet(responses, 0.75, '75network.dot','75network.dat',tree_dir='tree/',DEBUG=True)
Qnet.draw_Qnet('75network.dot',out_name = '75net.png')

#Step 5:
subprocess.Popen(["dot", '-Tpng', 'tree/P25.dot', '-o', 'decision_tree25.png'])
```

However, we will describe each step in detail here.

Step 1: Generating train and test csvs of sequences

First note that the sequences in **sequences.txt** are all 400 basis pairs in length. We will separate them into two csvs. One will be used for training and the other for testing. These csvs will be used to build the conditional inference trees. The **responses** variable designate every index we plan to build Ctrees for. We will build trees for all four hundred of them in this case. **test_ratio** describes the proportion of sequences that will be used in the testing set.

```
Qnet.fit_sequences('sequences.txt','train.csv','test.csv', test_ratio=0.2)
```

The csvs produced will contain a header from 0 to 399. The rows are comma separated letters of the sequences.

Step 2: Building the trees

Next, we will build the Ctrees. The arguments for **makeQNetwork** are the responses, defined previously, the two csvs produced in the previous step, and **tree_dir** a directory to place the trees in.

```
Qnet.makeQNetwork(responses, 'train.csv','test.csv',tree_dir='tree/', VERBOSE=True)
```

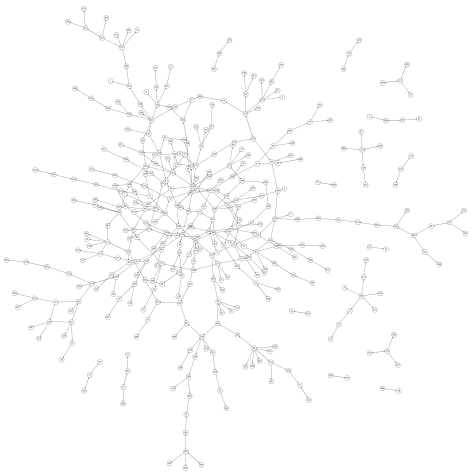
Pickle files of trees will be placed into the designated directory. These pickle files contain the information of the decision trees for each response.

Step 3 and Step 4: Visualizing the Qnetwork.

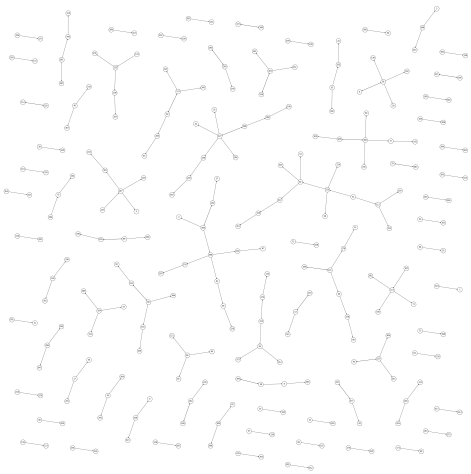
The next two steps examines the connections between responses(nodes). Between each pair of nodes, there is a measurement of feature importance. Given a threshold, 0.50 or 0.75, in this case, we parse out the connections that is greater than or equal to this threshold. Those connections will be the edges in our network graph. **connectQnet** will parse the trees for these connections and produce .dot and .dat files which contain representation of network. **draw_Qnet** takes the .dot representation of the qnetwork and draw it.

```
#Step 3:
Qnet.connectQnet(responses, 0.50, '50network.dot','50network.dat',tree_dir='tree/',DEBUG=True)
Qnet.draw_Qnet('50network.dot',out_name = '50net.png')

#Step 4:
Qnet.connectQnet(responses, 0.75, '75network.dot','75network.dat',tree_dir='tree/',DEBUG=True)
Qnet.draw_Qnet('75network.dot',out_name = '75net.png')
```



Qnetwork with threshold 0.50.



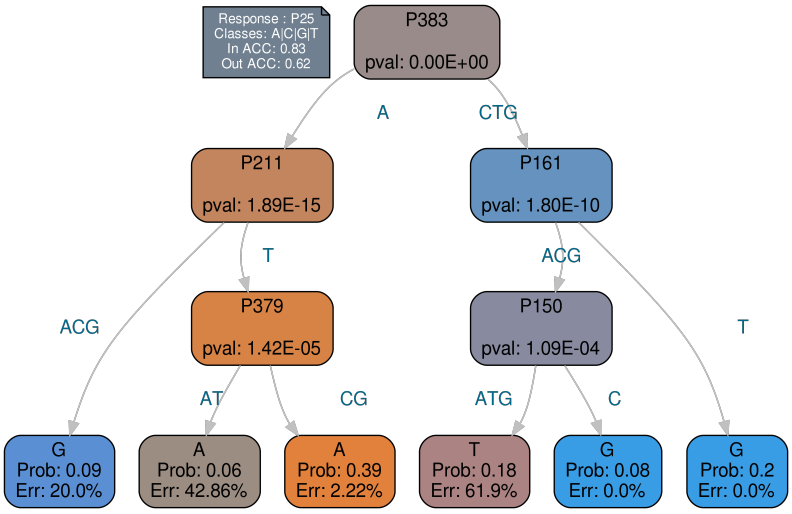
Qnetwork with threshold 0.75.

The above images are as expected. When the threshold for connection significance increases, fewer of the connections between nodes meet the threshold. Thus, the network with threshold 0.75 appears much less connected.

Step 5:Visualiing a single decision tree.

In step two, we produced a single dot file for each tree. These trees contains dot schemes for representing decision trees. The subprocess call in step 5 will draw the decision tree for position/node 25 of the Qnetwork.

```
subprocess.Popen(["dot", '-Tpng', 'tree/P25.dot', '-o', 'decision_tree25.png'])
```



The above visualization shows the decision making process of tree 25 representing the 26th (because of 0 indexing) basis pair in the DNA sequences.