lymph

Release 1.0.2

Roman Ludwig

Oct 15, 2021

CONTENT

This small pure-python package is an implementation of our model on metastatic spread of head & neck cancer (head & neck squamous cell carcinoma, to be precise). It is published under the MIT license on github and can hence be used freely. Here, in this documentation, we give you an insight in how to use this implementation but not how it works exactly. This has been layed out in much detail in our paper.

CHAPTER

ONE

INSTALLATION

This package is a pure python implementation, but it is not (yet) on PyPI, so it must be installed from a local directory. To do this, first clone the github repository

```
git clone https://github.com/rmnldwg/lymph.git
cd lymph
```

From here you can either use pip

```
pip install .
```

or the file `setup.py`

```
python setup.py
```

Note: You will need to have numpy and pandas installed and python 3.6 or higher.

In the future we plan to make the `lymph` package available on PyPI, so that it may be easier to install using *pip* or conda.

CHAPTER

TWO

GETTING STARTED

This package is meant to be a relatively simple-to-use frontend. The math is done under the hood and one does not need to worry about it a lot. Below are the things that are actually necessary.

2.1 Importing

This one is obvious.

[2]:
```
import lymph
```

2.2 Graph

The model is based on the assumption that one can represent the lymphatic system as a directed graph. Hence, the first thing to do is to define a graph that represents the drainage pathways of the lymphatic system aptly.

Here, this is done via a dictionary:

[3]:
```
graph = {'T': ['a', 'b'],
         'a': ['b', 'c'],
         'b': ['c'],
         'c': []}
```

Every key in this dictionary represents either a tumor - in which case its name must start with a `t` (upper- or lowercase) - or a lymph node level. The value of each of those nodes is a list of nodes it connects to. So, for example the primary tumor `T` in the `graph` above has directed arcs to `a` and `b`, while the LNL `c` does not have any outgoing connections.

We can simply create an instance of `System` using only this graph and let it report itself:

[4]:
```
simple_lymph_system = lymph.System(graph=graph)
simple_lymph_system.print_graph()
```

```
Tumor(s):
        T ---  0.0 % --> a
          +--  0.0 % --> b

LNL(s):
        a ---  0.0 % --> b
          +--  0.0 % --> c
        b ---  0.0 % --> c
```

The percentages between two nodes represents the probability (rate) that metastatic spread occurs along it. In the case of the tumor spreading to LNL a we call this probability *base probability (rate)* $\tilde{b}_a$. For the spread between lymph node levels, we call it *transition probability (rate)*, e.g. $\tilde{t}_{ab}$. The difference to the base probability (rate) is that it only plays a role if the parent LNL is already involved with metastases, while the tumor always spread, of course.

We can change these probability (rates) with a function called `set_theta()`. The only argument to this function is a list or array of these values in the order in which they appear when we call `print_graph()`:

[5]:
```
simple_lymph_system.set_theta([0.1, 0.2, 0.4, 0.25, 0.05])
simple_lymph_system.print_graph()
```

```
Tumor(s):
        T --- 10.0 % --> a
          +-- 20.0 % --> b

LNL(s):
        a --- 40.0 % --> b
          +-- 25.0 % --> c
        b ---  5.0 % --> c
```

Reversely, we can also read them out:

[6]:
```
probabilities = simple_lymph_system.get_theta()
print(probabilities)
```

```
[0.1  0.2  0.4  0.25 0.05]
```

2.3 Diagnostic Modalities

To ultimately compute the likelihoods of observations, we need to fix the sensitivities and specificities of the obtained diagnoses. And since we might have multiple diagnostic modalities available, we need to tell the system which of them comes with which specificity and sensitivity. We do this by creating a dictionary of specificity/sensitivity pairs:

[7]:
```
spsn_dict = {"MRI": [0.63, 0.81],
             "PET": [0.86, 0.79]}
```

Now we can pass this to the system using the `set_modalities` function.

[8]:
```
simple_lymph_system.set_modalities(spsn_dict=spsn_dict)
```

However, mostly this is going to be done automatically when loading the data and providing the dictionary along with the dataset.

2.4 Data / Observations

To compute the likelihood of a set of probability (rates) given a patient cohort we need such a patient cohort, of course. We can provide it to the system in the form of a `pandas DataFrame`. Here is an example:

[9]:
```
import pandas as pd

dataset = pd.read_csv("_data/example.csv", header=[0,1])
dataset
```

[9]:
```
       Info pathology
    T-stage         I II III IV
0     early         1  0   0  0
1     early         0  1   0  0
2     early         0  1   0  0
3     early         0  1   0  0
4     early         0  1   0  0
..      ...       ... ..  .. ..
142   early         0  0   0  0
143   early         0  0   0  0
144   early         0  0   0  0
145   early         0  0   0  0
146   early         0  0   0  0

[147 rows x 5 columns]
```

Note that this data has two header-rows, defining not only the individual column's content, but also to which overarching category they belong. The "Info" category plays a special role here along with its sub-category "T-stage". It will later tell the system which time prior to use according to a dictionary of these distributions.

The "pathology" section denotes that this dataset contains observations from a pathologic diagnostic modality (neck dissections in this case). How this is termed is irrelevant, as we will be telling the system what to look for. Import is, however, that - if we had multiple diagnostic modalities - they all contain a column for each lymph node level in the system we have set up. Obvioulsy, this dataset here does not match the system set up earlier, so let's fix that.

[10]:
```
graph = {'T'  : ['I', 'II', 'III', 'IV'],
         'I'  : ['II'],
         'II' : ['III'],
         'III': ['IV'],
         'IV' : []}

example_system = lymph.System(graph=graph)
example_system.print_graph()
```

```
Tumor(s):
        T ---  0.0 % --> I
          +--  0.0 % --> II
          +--  0.0 % --> III
          +--  0.0 % --> IV

LNL(s):
        I ---  0.0 % --> II
        II ---  0.0 % --> III
        III ---  0.0 % --> IV
```

To feed the dataset into the system, we use the appropriate function. It also takes as an argument all the different T-stages that we want to consider. What the system does here is creating a $\mathbf{C}$ matrix for every T-stage we provide here. Keep in mind that the T-stages in this list must actually occur in the database.

Similarly, we can provide our specificity/sensitivity pairs for each diagnsotic modality directly here. In this case too the keys of the dictionary must be columns in the dataset.

[12]:
```
only_pathology = {"pathology": [1., 1.]}

example_system.load_data(dataset,
```

(continues on next page)

(continued from previous page)

```
                        t_stage=["early"],
                        spsn_dict=only_pathology,
                        mode="HMM")
```

Lastly, the `mode` parameter determines which model should be used. We have implemented both the Bayesian network (`mode = "BN"`) from (Pouymayou et al., 2019) and the hidden Markov model from our work (`mode = "HMM"`). In case the Bayesian network is chosen, the parameter `t_stage` has no effect.

2.5 Time prior

The last ingerdient to set up (at least when using the hidden Markov model) would now be the time prior. Since this dataset contains only early T-stage patients the exact shape does not matter too much, as long as it is "reasonable". If we also had late T-stage patients in the cohort, we would need to think about how the two time priors relate to each other.

For now we are going to use binomial distributions for this. Their shape makes intuitive sense: Since the time prior $p_T(t)$ is a distribution over the probability of diagnosing a patient after t time steps, given his T-stage T we would expect that a very early detection of the cancer is similarly unlikely as a very late one.

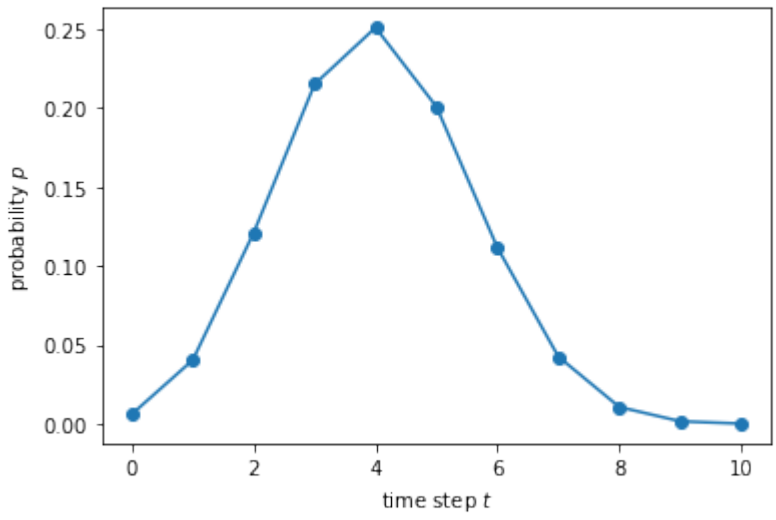
As mentioned, we need to put a distribution for each distinct T-stage in our cohort into a dictionary:

[14]:
```
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt

t_max = 10
time_steps = np.arange(t_max+1)
p = 0.4

early_prior = sp.stats.binom.pmf(time_steps, t_max, p)

plt.plot(time_steps, early_prior, "o-");
plt.xlabel("time step $t$");
plt.ylabel("probability $p$");
```



[15]:
```
time_prior_dict = {}
time_prior_dict["early"] = early_prior
```

2.6 Likelihood

With everything set up like this, we can compute the likelihood of seeing the above dataset given a set of base and transition probability (rates).

[16]:
```
test_probabilities = [0.02, 0.24, 0.03, 0.2, 0.23, 0.18, 0.18]

llh = example_system.likelihood(test_probabilities,
                                t_stage=["early"],
                                time_prior_dict=time_prior_dict,
                                mode="HMM")

print(f"log-likelihood is {llh:.2f}")
```

```
log-likelihood is -331.09
```

[17]:
```
example_system.print_graph()
```

```
Tumor(s):
        T ---  2.0 % --> I
          +-- 24.0 % --> II
          +--  3.0 % --> III
          +-- 20.0 % --> IV

LNL(s):
        I --- 23.0 % --> II
        II --- 18.0 % --> III
        III --- 18.0 % --> IV
```

From here it is up to the user what to do with this quantity. Most *likely* though, one would want to perform MCMC sampling with this.

CHAPTER

THREE

DETAILED API

The human lymph system (or rather parts of it) are modelled as directed graphs here. Hence, a `System` consists of multiple `Node` and `Edge` instances, which are represented by a python class each.

3.1 Lymph system

class lymph.**System**(*graph={}*)

Class that describes a whole lymphatic system with its lymph node levels (LNLs) and the connections between them.

Parameters **graph** (`dict`) – For every key in the dictionary, the `system` will create a `node` that represents a binary random variable. The values in the dictionary should then be the a list of names to which `edges` from the current key should be created.

combined_likelihood(*theta*, *t_stage=['early', 'late']*, *time_prior_dict={}*, *T_max=10*)

Likelihood for learning both the system's parameters and the center of a Binomially shaped time prior.

Parameters

- **theta** (`ndarray`) – Set of parameters, consisting of the base probabilities b (as many as the system has nodes), the transition probabilities t (as many as the system has edges) and - in this particular case - the binomial parameters for all but the first T-stage's time prior.
- **t_stage** (`List[str]`) – keywords of T-stages that are present in the dictionary of C matrices. (default: `["early", "late"]`)
- **time_prior_dict** (`dict`) – Dictionary with keys of T-stages in `t_stage` and values of time priors for each of those T-stages.
- **T_max** (`int`) – maximum number of time steps.

Return type `float`

Returns The combined likelihood of observing patients with different T-stages, given the spread probabilities as well as the parameters for the later (except the first) T-stage's binomial time prior.

find_edge(*startname*, *endname*)

Finds and returns the edge instance which has a parent node named `startname` and ends with node `endname`.

Return type `Optional[Edge]`

find_node(*name*)

Finds and returns a node with name `name`.

Return type `Optional[Node]`

`get_graph()`
Lists the graph as it was provided when the system was created

Return type `dict`

`get_theta()`
Returns the parameters currently set. It will return the transition probabilities in the order they appear in the graph dictionary. This deviates somewhat from the notation in the paper, where base and transition probabilities are distinguished as probabilities along edges from primary tumour to LNL and from LNL to LNL respectively.

Return type `List[float]`

`likelihood`(*theta*, *t_stage=[1, 2, 3, 4]*, *time_prior_dict={}*, *mode='HMM'*)
Computes the likelihood of a set of parameters, given the already stored data(set).

Parameters

- **theta** (`ndarray`) – Set of parameters, consisting of the base probabilities b (as many as the system has nodes) and the transition probabilities t (as many as the system has edges).
- **t_stage** (`List[int]`) – List of T-stages that should be included in the learning process. (default: `[1,2,3,4]`)
- **time_prior_dict** (`dict`) – Dictionary with keys of T-stages in `t_stage` and values of time priors for each of those T-stages.
- **mode** (`str`) – `"HMM"` for hidden Markov model and `"BN"` for Bayesian network. (default: `"HMM"`)

Return type `float`

Returns The log-likelihood of a parameter sample.

`list_edges()`
Lists all edges of the system with its corresponding start and end states

Return type `List[Edge]`

`load_data`(*data*, *t_stage=[1, 2, 3, 4]*, *spsn_dict={'path': [1.0, 1.0]}*, *mode='HMM'*)
Generates the matrix C that marginalizes over multiple states for data with incomplete observation, as well as how often these obser- vations occur in the dataset. In the end the computation $\mathbf{p} = \boldsymbol{\pi} \cdot \mathbf{A}^t \cdot \mathbf{B} \cdot \mathbf{C}$ results in an array of probabilities that can - together with the frequencies f - be used to compute the likelihood. This also works for the Bayesian network case: $\mathbf{p} = \mathbf{a} \cdot \mathbf{C}$ where a is an array containing the probability for each state.

Parameters

- **data** (`DataFrame`) – Contains rows of patient data. The columns must include the T-stage and at least one diagnostic modality.
- **t_stage** (`List[int]`) – List of T-stages that should be included in the learning process. (default: `[1,2,3,4]`)
- **spsn_dict** (`Dict[str, List[float]]`) – Dictionary of specificity s_P and s_N (in that order) for each observational/diagnostic modality. (default: `{"path": [1., 1.]}`)
- **mode** (`str`) – `"HMM"` for hidden Markov model and `"BN"` for Bayesian network. (default: `"HMM"`)

`obs_prob`(*diagnoses_dict*, *log=False*)
Computes the probability to see certain diagnoses, given the system's current state.

Parameters

- **diagnoses_dict** (Dict[str, List[int]]) – Dictionary of diagnoses (one for each diagnostic modality). A diagnose must be an array of integers that is as long as the the system has LNLs.
- **log** (bool) – If True, the log probability is computed. (default: False)

Return type float

Returns The probability to see the given diagnoses.

print_graph()
Print info about the structure and parameters of the graph.

risk(*inv*, *obs*, *time_prior=[]*, *mode='HMM'*)
Computes the risk for involvement (or no involvement), given some observations and a time distribution for the Markov model (and the Bayesian network).

Parameters

- **inv** (ndarray) – Pattern of involvement that we want to compute the risk for. Values can take on the values 0 (*negative*), 1 (*positive*) and None of we don't care if this is involved or not.
- **obs** (Dict[str, ndarray]) – Holds a diagnose of similar kind as inv for each diagnostic modality. An incomplete diagnose can be filled with None.
- **time_prior** (List[float]) – Discrete distribution over the time steps. Must hence sum to 1.
- **mode** (str) – "HMM" for hidden Markov model and "BN" for Bayesian network. (default: "HMM")

Return type float

Returns The risk for the involvement of interest, given an observation.

set_modalities(*spsn_dict={'path': [1.0, 1.0]}*)
Given some 2x2 matrices for each diagnostic modality based on their specificity and sensitivity, compute observation matrix $\mathbf{B}$ and store the details of the diagnostic modalities.

set_state(*newstate*)
Sets the state of the system to newstate.

set_theta(*theta*, *mode='HMM'*)
Fills the system with new base and transition probabilities and also computes the transition matrix A again, if one is in mode "HMM".

Parameters

- **theta** (ndarray) – The new parameters that should be fed into the system. They all represent the transition probabilities along the edges of the network and will be set in the order they appear in the graph dictionary. As mentioned in the get_theta() function, this includes the spread probabilities from the primary tumour to the LNLs, as well as the spread among the LNLs.
- **mode** (str) – If one is in "BN" mode (Bayesian network), then it is not necessary to compute the transition matrix A again, so it is skipped. (default: "HMM")

trans_prob(*newstate*, *log=False*, *acquire=False*)
Computes the probability to transition to newstate, given its current state.

Parameters

- **newstate** (List[int]) – List of new states for each LNL in the lymphatic system. The transition probability t will be computed from the current states to these states.
- **log** (bool) – if True, the log-probability is computed. (default: False)
- **acquire** (bool) – if True, after computing and returning the probability, the system updates its own state to be newstate. (default: False)

Return type float

Returns Transition probability t.

unparametrized_epoch(*t_stage=[1, 2, 3, 4]*, *time_prior_dict={}*, *T=1.0*, *scale=0.01*)

An attempt at unparametrized sampling, where the algorithm samples A from the full solution space of row-stochastic matrices with zeros where transitions are forbidden.

Parameters

- **t_stage** (List[int]) – List of T-stages that should be included in the learning process. (default: [1,2,3,4])
- **time_prior_dict** (dict) – Dictionary with keys of T-stages in t_stage and values of time priors for each of those T-stages.
- **T** (float) – Temperature of the epoch. Can be reduced from a starting value down to almost zero for an annealing approach to sampling. (default: 1.)

Return type float

Returns The log-likelihood of the epoch.

3.2 Edge

Represents a lymphatic drainage pathway and therefore are spread probability.

class lymph.**Edge**(*start*, *end*, *t=0.0*)

Class for the connections between lymph node levels (LNLs) represented by the Node class.

Parameters

- **start** (*Node*) – Parent node
- **end** (*Node*) – Child node
- **t** (float) – Transition probability in case start-Node has state 1 (microscopic involvement).

report()

Just quickly prints infos about the edge

3.3 Node

Represents a lymph node level (LNL) or rather a random variable associated with it. It encodes the microscopic involvement of the LNL and - if involved - might spread along outgoing edges.

class lymph.**Node**(*name*, *state=0*, *typ=None*)

Class for lymph node levels (LNLs) in a lymphatic system.

Parameters

- **name** (str) – Name of the node

- **state** (int) – Current state this LNL is in. Can be in {0, 1}
- **typ** (Optional[str]) – Can be either *"lnl"*, *"tumor"* or *None*. If it is the latter, the type will be inferred from the name of the node. A node starting with a *t* (case-insensitive), then it will be a tumor node and a lymph node levle (lnl) otherwise. (default: *None*)

bn_prob(*log=False*)

Computes the conditional probability of a node being in the state it is in, given its parents are in the states they are in.

Parameters **log** (bool) – If True, returns the log-probability. (default: False)

Return type float

Returns The conditional (log-)probability.

obs_prob(*obs*, *obstable=array([[1.0, 0.0], [0.0, 1.0]])*, *log=False*)

Compute the probability of observing a certain diagnose, given its current state.

Parameters

- **obs** (int) – Diagnose/observation for the node.
- **obstable** (ndarray) – 2x2 matrix containing info about sensitivity and specificty of the observational/diagnostic modality from which *obs* was obtained.
- **log** (bool) – If True, method returns the log-prob.

Return type float

Returns The probability of observing the given diagnose.

report()

Just quickly print infos about the node.

trans_prob(*log=False*)

Computes the transition probabilities from the current state to all other possible states.

Parameters **log** (bool) – If True method returns the log-probability. (default: False)

Return type float

Returns

The transition probabilities from current state to all two other states.

CHAPTER

FOUR

LICENSE

CHAPTER

FIVE

INDEX & SEARCH

- genindex
- search

INDEX