gfp_gaussian_process

Likelihood calculation and predictions of 1 dimensional genealogy is re-implementaion of the python code: https://github.com/fioriathos/new_protein_project.

Usage

1 Compile

1.1 Compile locally

The following two libraries are needed:

- nlopt (for minimization)
  - see https://nlopt.readthedocs.io/en/latest/#download-and-installation
- Eigen (for linear algebra)
  - see http://eigen.tuxfamily.org/index.php?title=Main_Page

Make sure the correct paths to the two libraries are set in the `Makefile`. Currently both are assumed to be located in the home directory. Then, compile with:

`cd src; make local`

1.2 Compile on cluster

1. Install nlopt

- Will install nlopt in home-directory with static linking. You can change that via `DCMAKE_INSTALL_PREFIX`, but make sure to adjust the makefile accordingly!

```
ml CMake
wget https://github.com/stevengj/nlopt/archive/v2.6.2.tar.gz
tar -xf v2.6.2.tar.gz
cd nlopt-2.6.2/
cmake -DCMAKE_INSTALL_PREFIX=~/nlopt -DBUILD_SHARED_LIBS=OFF .
make
make install
```

2. Compile
   Run `cd src; make cluster`. This will run `ml GCC/8.3.0; ml Eigen/3.3.7` as well as the compile command! Note, that the modules remain loaded after compilation.

2 Run

`cd bin`
`./gfp_gaussian [-options]` with following options:

```
-h, --help                this help message
-i, --infile              (required) input data file
-b, --parameter_bounds    (required) file setting the type, step, bounds of the parameters
-c, --csv_config          file that sets the colums that will be used from the input file
-l, --print_level         print level >=0, default=0
-o, --outdir              specify output direction and do not use default
-t, --tolerance           absolute tolerance of maximization between optimization steps, default=1e-1
-m, --maximize            run maximization
-s, --scan                run 1d parameter scan
-p, --predict             run prediction
```

Example:

```
./gfp_gaussian -c csv_config.txt -b parameter_min.txt -i ../data/simulation_gaussian_gfp.csv -o out/ -l 1 -t 1e-2 -m -p
```

2.1 Required arguments

- `infile` sets the input file that contains the data, eg as given by MOMA
- `parameter_bounds` sets the file that defines the parameter space

2.1.1 Input file

The input file is assumed to fullfil the following:

- the data points of a cell appear as consecutive rows and are in the correct order with repect to time
- the data set has to include all columns that are set via the `csv_config` file, i.e. time_col, length_col, fp_col
- the cells can be uniquely identified via the tags provided via `parent_tags` and `cell_tags` and each mother cell has at most 2 daughter cells. If that is not the case, the `parent_tags` and `cell_tags` are not sufficient and a warning will be printed.
- In order to estimate the initial covariance matrix, the data set needs to contain at least (!) 2 cells.

2.1.2 Parameter file

Syntax for free, bound, fixed (in that order) parameters

- parameter = init, step
- parameter = init, step, lower, upper
- parameter = init

Example:

```
mean_lambda = 0.01, 1e-3
gamma_lambda = 0.01, 1e-3, 1e-4, 0.05
var_lambda = 1e-07
```

ALL parameters are restricted to positive numbers by default avoiding unphysical/meaningless parameter ranges. By setting the lower bound in the parameter file, one can overwrite the lower bounds of the parameters.

The step value is used for the 1d scan to discretize the interval set by lower and upper. During the maximization this will be the initial step size. From nlopt doc:
"For derivative-free local-optimization algorithms, the optimizer must somehow decide on some initial step size to perturb x by when it begins the optimization. This step size should be big enough that the value of the objective changes significantly, but not too big if you want to find the local optimum nearest to x."

2.2 Optional arguments

- `csv_config` sets the file that contains information on which columns will be used from the input file
- `print_level=0` supresses input of the likelihood calculation, `1` prints every step of the maximization/scan/error bar calculation. This is purely meant for debugging!
- `tolerance` sets the stopping critirium by setting the tolerance of maximization: Stop when an optimization step changes the function value by less than tolerance. By setting very low tolerances one might encounder rounding issues, in that case the last valid

step is taken and a warning is printed to stderr.
- `outdir` overwrites default output directory, which is (given the infile `dir/example.csv/`) `dir/example_out/`
- run modes

2.2.1 Csv_config file

The following settings define how the input file will be read. Default values in brackets.

- time_col (time_sec): column from which the time is read
- rescale_time (60): factor by which time will be devided before the anything is run. This can be used to use a different time unit for the input than for the model paramters.
- length_col (length_um): column from which the length of the cell is read
- length_islog (false): indicates if the cell length in the data file is in logscale (true) of not (false)
- fp_col (gfp_nb): column from which the intensity is read
- delm (,): delimiter between columns, probably ',' or ';'
- cell_tag (date, pos, gl, id): columns that will make up the unique cell id, endings like .0 .00 etc of numeric values will be removed
- parent_tags (date, pos, gl, parent_id): columns that will make up the unique cell id of the parent cell, endings like .0 .00 etc of numeric values will be removed

2.2.2 Run modes

- `m (maximize), s(scan), p(predict)` will run the respective task. In case `maximize` and `predict` is set, the estimated paramters after the maximization will be used for the prediction. Those paramters that are fixed are of course not effected.
- the 1d parameters scans will calculate the likelihood for the 1d ranges set by the parameter_bound file. Note, only "bound" parameters will be scaned.
-

3 Model parameters

The 2 OU processes are descibed with a mean value (thus the mean growth/production rate), a gamma parameter determining how fast the process is driven towards its mean after a deviation, and a variance that scales the noise term. Thus we have the following parameters including the bleaching rate of the fp, beta:

- Growth rate fluctualtions params:
  - mean_lambda
  - gamma_lambda
  - var_lambda
- gfp fluctuation params:
  - mean_q
  - gamma_q
  - var_q
  - beta

Additionally, the lenth of the cell and the gfp is effected by measurement noise

- Measurement noise:
  - var_x
  - var_g

Finally, asymmentric cell division is modeled via two variances of gaussians

- cell division:
  - var_dx
  - var_dg

4 Output

4.1 Maximization

- Will create 2 files: one for the maximization process and one for the final estimations
- The files (given the input file `example.csv`) are named as follows: `example_f<free>_b<bounds>.csv` and `example_f<free>_b<bounds>_final.csv`, where `<free>` lists the variable via the index as e.g. printed when the code is run and `<bounds>` lists the bound parameters in the same way. Example: `example_f034_b129.csv`, and `example_f034_b129_final.csv`
- The first file contains the parameter settings at the top 12 lines and all steps of the likelihood maximization
- The second file contains the parameter settings at the top 12 line including a column with the final parameters (i.e. estimated via log likelihood maximization and init value for free parameters) and the estimated error for the estimated paramters via a hessian matrix (**beta version!!!**). The hessian is calculted using a range of finit-differnences that are set relative to the value of the respective paramter. I.e. epsilon=1e-2 corresponds to 1% of each paramterer is used for the hessian matrix estimation. Finally, the maximized log likelihood is written.

4.2 1D scan

- Will create a file for each parameter containing the parameter settings at the top 12 lines and all calculated likelihoods of the scan
- The file (given the input file `example.csv`) is named as follows: `example_<parameter>.csv`, where `<parameter>` is the paramter that is scaned

4.3 Predictions

- Will create a file for each parameter containing the parameter settings at the top 12 lines and ...
    - the 4 mean quanties of x, g, l/lambda, q
    - the upper triangle of the covariance matrix
- ... of each time point for each cell in the same order as the input file

Notes

TODO:

- [x] prepare for cluster
- [x] write new simulation including asymmetric cell division and tree structure?
- [ ] autocorrelation
- [ ] use log of parameters

Technical Notes

Log-Likelihood maximization with multiple cell trees

The log-likelihoods of all cell trees are added and the summed log-likelihood is then maximized. This of course implies that the parameters are the same for all cell trees.

Cell division in backwards direction

In forward direction the first time point distributions of daughter cells are calculated from the mother cell (including asymmetric cell division. In the backward direction, we might have two daughter cells that determine the distribution of the mother cell. In that case, two mother cells are calculated and the distribution of the two are multiplied to obtain a single distribution for the last time point of the mother cell.

Likelihhod and prediction calculation along cell tree

- function for likelihood and predictions for the single cells are applies recursively to go up/down the tree

- every cell is accessed once
- every cell is accessed after its parent(down direction)/daughter cells(up direction)

```
    /* applies the function func to the cell cell and the other cells in the genealogy
    * such that the parent cell has already been accessed when the function is applied
    * to the cell.
    *
    * Example (number implies the order in which cell is accessed)
    * __________________________________________________

              1             |
            /   \           |
           2     5          |
         /   \     \        |
        3     4     6       V

    * __________________________________________________
    */
void prediction_forward(const std::vector<double> &params_vec, std::vector<MOMAdata> &cells){
    std::vector<MOMAdata *> p_roots = get_roots(cells);

    for(int i=0; i<p_roots.size(); ++i){
        prediction_forward_recr(params_vec,  p_roots[i]);
    }
}

    /* applies the function func to the cell and the other cells in the genealogy
    * such that the daughter cells has already been accessed when the function is applied
    * to the cell.
    *
    * Example (number implies the order in which cell is accessed)
    * __________________________________________________

              6             ^
            /   \           |
           3     5          |
         /   \     \        |
        1     2     4       |

    * __________________________________________________
    */
void prediction_backward(const std::vector<double> &params_vec, std::vector<MOMAdata> &cells){
    std::vector<MOMAdata *> p_roots = get_roots(cells);

    for(int i=0; i<p_roots.size(); ++i){
        prediction_backward_recr(params_vec,  p_roots[i]);
    }
}
```

Minimizer

- nlopt
- Note, `total_likelihood` return -tl (negative total log_likelihood). Thus, maximizing the log_likelihod, becomes minimization.
- the wrapper `double total_likelihood(const std::vector<double> &params_vec, std::vector<MOMAdata> &cells);` meant for direct calculation without minimization returns just the log_likelihood

```
double minimize_wrapper(double (*target_func)(const std::vector<double> &x, std::vector<double> &grad, void *p),
                        std::vector<MOMAdata> &cells,
                        Parameter_set &params,
                        double tolerance,
                        bool &found_min)
```

Current minimizer: LN_BOBYQA

- from doc: "This is an algorithm derived from the BOBYQA subroutine of M. J. D. Powell, converted to C and modified for the NLopt stopping criteria. BOBYQA performs derivative-free bound-constrained optimization using an iteratively constructed quadratic approximation for the objective function." See also http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf for details.