ACE build passing

Table of contents

Introduction

ACE is a software package designed to quickly and accurately infer Ising or Potts models based on correlation data from a variety of biological and artificial systems. This software makes use of the **A**daptive **C**luster **E**xpansion (ACE) algorithm.

Given a set of correlation data or sequence input in FASTA format, ACE will produce a Ising or Potts model that reproduces the input correlations to within the expected error due to finite sampling.

NOTE: Mathematical expressions through MathJax are currently not supported on GitHub.

To see these expressions rendered properly, please see `README.pdf`.

Installation

Download and unzip the package, then run the following commands in the terminal from the new directory:

```
$ ./configure
$ make
```

If you'd like to be able to run the program from any directory, you can then enter:

```
$ make install
```

Required Input

Running the algorithm requires a set of correlations as input, to be computed from your data.

As an example, let's consider a system of N variables described by the configuration $\underline{x} = \{x_1, x_2, \ldots, x_N\}$, with each variable x_i taking one of q_i possible values, $x_i \in \{1, 2, \ldots, q_i\}$. From a set of B observations of the system, we can compute the frequency of each variable as well as the pairwise correlations,

$$p_i(a) = \frac{1}{B}\sum_{k=1}^{B}\delta(x_i, a)\,,$$

$$p_{ij}(a,b) = \frac{1}{B}\sum_{k=1}^{B}\delta(x_i, a)\delta(x_j, b)\,.$$

Here δ represents the Kronecker delta function. These correlations should be saved in a file ending with the extension `.p`, in the following format:

$$
\begin{array}{l}
p_1(1)\, p_1(2) \ldots p_1(q_1 - 1) \\
p_2(1)\, p_2(2) \ldots p_2(q_2 - 1) \\
\ldots \\
p_N(1)\, p_N(2) \ldots p_N(q_N - 1) \\
p_{1,2}(1,1)\, p_{1,2}(1,2) \ldots p_{1,2}(1,q_2 - 1)\, p_{1,2}(2,1)\, p_{1,2}(2,2) \ldots p_{1,2}(q_1 - 1, q_2 - 1) \\
p_{1,3}(1,1) \ldots
\end{array}
$$

In other words, the first N lines of the file record the frequency that each state is observed at each site, and the next $N(N-1)/2$ lines record the pairwise correlations. Note that, because $\sum_{a=1}^{q_i} p_i(a) = 1$, the frequency (and corresponding pair correlations) for one state at each site need not be specified explicitly.

These values should be given in floating point or scientific format, with whitespace (e.g. `'\t'`) between successive values and a newline character (`'\n'`) at the end of each line. In order for the correlations to be read in properly, there should be **no** whitespace between the final correlation value and the newline character on each line.

For examples, see the `examples/` directory. Instructions on how to automatically generate correlations from a sequence alignment in FASTA format (and others) can be found in the Matlab file in the `scripts/` directory. The correlations can also be prepared through a set of Python scripts, `scripts/ACEtools.py`, which also includes useful auxiliary functions.

Running the program

Here we show a simple example of how to run the program and interpret the output, using a set of sample data for the HIV protein p7. Full explanations for the possible options are given here.

Running ACE

We begin running the ACE algorithm on the example p7 dataset with the command:

```
$ ./bin/ace -d examples -i p7 -o p7-out -g2 0.0002 -b 4130
```

This creates two new files, `examples/p7-out.sce` and `examples/p7-out.j`, which record general output on the inference procedure and the current inferred Potts parameters, respectively.

Output from the first file, `examples/p7-out.sce`, should appear something like the following:

```
8.463532e-04 1.917385e+01 9.964734e+00 1.309919e+02 1.186601e+01 4 3321 364
8.060507e-04 1.911941e+01 1.019924e+01 1.334989e+02 1.186352e+01 4 3343 373
7.676673e-04 2.298425e+01 1.194769e+01 1.659755e+02 1.186115e+01 4 3353 380
7.311117e-04 2.550650e+01 1.277885e+01 1.768295e+02 1.185510e+01 4 3399 390
6.962969e-04 1.842427e+01 9.891297e+00 1.219612e+02 1.185371e+01 4 3442 406
...
```

These columns represent, respectively: the current value of the threshold θ, error on the one-point correlations ϵ_{p1}, error on the pairwise correlations ϵ_{p2}, normalized maximum error $\epsilon_{\max}$, current estimate of the entropy S, maximum cluster size, total number of clusters in the expansion, and the number of selected clusters (i.e. those for which $|\Delta S| > \theta$).

The inferred Potts parameters in the second file, `examples/p7-out.j`, are output in the same format as the input correlations, as shown above. In this case, the first N lines record the Potts fields $h_i(a)$, and the following $N(N-1)/2$ lines record the couplings $J_{ij}(a,b)$.

The final line of `examples/p7-out.sce` should then appear something like:

```
1.338016e-05 3.641289e-01 1.674574e-01 9.498217e-01 1.169706e+01 6 20293 4773
```

The error for the Potts parameters is low (the error terms $\epsilon_{p1}, \epsilon_{p2}, \epsilon_{\max} < 1$), but we can follow this initial inference step by running the **M**onte **C**arlo (MC) learning algorithm to ensure convergence. This is particularly useful when convergence is difficult to obtain in the cluster algorithm alone. Typically we find that MC learning is more likely to be successful when the entropy has nearly converged (see column 6 in `examples/p7-out.sce`).

Running the MC learning algorithm QLS

We now run the MC algorithm on the output we previously obtained from ACE, using the command:

```
$ ./bin/qls -d examples -c p7 -i p7-out -o p7-out-learn -g2 0.0002
-b 4130
```

This creates two additional output files, `examples/p7-out-learn.fit` and `examples/p7-out-learn.j`, which record progress on the MC learning procedure and the current refined Potts parameters, respectively.

Output from the first file, `examples/p7-out-learn.fit`, should appear something like the following:

```
1 1.877327e-01 1.266454e-01 1.303780e+00 1.900000e+00
2 1.976668e-01 1.284943e-01 1.329022e+00 3.610000e+00
3 1.956319e-01 1.276321e-01 1.340715e+00 6.859000e+00
4 1.959972e-01 1.282958e-01 1.279392e+00 1.303210e+01
5 2.073654e-01 1.295238e-01 1.377221e+00 2.476099e+01
...
```

These columns represent, respectively: the current iteration, error on the one-point correlations ϵ_{p1}, error on the pairwise correlations ϵ_{p2}, normalized maximum error $\epsilon_{\max}$, and the maximum size of the weight parameter used in the MC learning update step. Note that the error is slightly different in this case than at the end of the cluster algorithm. This is because, by default, the MC learning algorithm computes the correlations using a larger number of samples.

After about 15 iterations the MC learning algorithm should converge and the program will terminate. The Potts parameters recorded in the second file, `examples/p7-out-learn.j`, now specify a model that accurately recovers the input correlations to within fluctuations expected due to finite sampling.

Verifying the output with QGT

If the input correlations are generated using the matlab script included in this package (for instructions, see the script itself, which lies in the `scripts/` directory), auxiliary measurements such as higher order correlations can be checked comprehensively. By default, this routine compares the one- and two-point correlations for the model and data, as well as the probability $P(k)$ of observing k differences between sampled configurations and the “consensus” (determined from input read in to the program). It is also possible to compute the three-point correlations, the energy distribution, and a set of sample configurations.

This routine can be run using, for example:

```
$ ./bin/qgt -d examples -c p7 -m p7 -w p7 -i p7-out -o p7-out-fit
-g2 0.0002 -b 4130
```

This program also gives the RMS error and Pearson correlation between the model and data correlations.

Note that the output correlations from this program include not just the input ones, but also the ones corresponding to the “gauged” states, which are implicit in the input data. We also note that by default the output of small three-point correlations is trimmed in order to prevent generating extremely large files. For more information, see the options below.

Troubleshooting

On difficult data sets (for example, systems of very large size, those with many states, and/or high variability), ACE may be more slow to converge. Below are a few common potential problems and suggestions for how to fix them.

The one- and two-point error terms have converged ($\epsilon \leq 1$), but the maximum error remains > 1. Obtaining a normalized maximum error < 1 is the most stringent statistical check of the inferred model performed by the ACE and QLS routines. It is possible to obtain a very good generative model of the data even in cases where this error is larger than 1 – use QGT to verify an acceptable fit.

The entropy is oscillating. Large oscillations of the entropy can be observed when whole collections of variables strongly interact. One common source is strongly-correlated gaps at the beginning and end of sequences from protein families. In such cases, filling in gaps while computing the correlations, using stronger compression, or increasing the regularization strength can help to reduce oscillations and improve convergence.

ACE slows down as the size of clusters increases. This can occur if the number of states is very large. Stronger compression can allow the cluster expansion algorithm to proceed further. In addition, QLS can often converge rapidly after the network of strong interactions has been inferred by ACE, even if the errors are high and ACE has not yet advanced to a low value of the threshold (see the lattice protein model here for an example).

Additional questions? Please contact us for more information or advice on dealing with difficult data sets.

Command line options

Options for all programs

- `-d` gives the path to the directory where data files are located, and where output will be written (default: "." (current directory))
- `-i` specifies the name of the input file (excluding the extension, default: "input")
- `-o` specifies the name of the output file (excluding the extension, default: "output")
- `-v` enables verbose output
- `-b` tells the program how many samples were used to generate the input correlations, so that the expected error in the correlations due to finite sampling can be estimated (default: 1000)
- `-mcb` gives the number of Monte Carlo steps used to estimate the inference error (default: 40000 (ace), 800000 (qls, qgt))
- `-mcr` gives the number of independent Monte Carlo trajectories to use when estimating the inference error (default: 1)
- `-g2` sets the L_2-norm regularization strength (note that a natural value for this parameter is $1/B$, where B is the number of samples used to generate the input correlations – for contact prediction, it may be best to use strong regularization ≈ 1, regardless of the number of samples, default: 0)
- `-ag` automatically sets the L_2-norm regularization strength equal to $1/B$, using the number of samples B passed with the `-b` option
- `-gi` enable the alternate gauge-invariant form of the L_2 regularization for couplings (see here for details)

Additional ACE options

- `-kmin` sets the minimum cluster size required before the program will terminate (default: 0)
- `-kmax` sets the maximum cluster size; the program terminates automatically after a cluster of this size is created (default: none)
- `-t` specifies a single value of the threshold θ at which the algorithm will run, then

exit
- `-tmax` specifies the maximum (starting) value of the threshold (default: 1)
- `-tmin` specifies the minimum allowed value of the threshold; the program terminates automatically after θ falls below this minimum value (default: 1e-10)
- `-ts` specifies the logarithmic step size to for between successive values θ, through $\theta_{i+1} = \theta_i / \theta_{\rm step}$ (default: 1.05)
- `-r` enables the expansion of the entropy S around a mean-field reference entropy S_0, which may be helpful in particular for inferring models described by dense networks of weak interactions (note: works only if all variables are binary)
- `-g0` sets the L_0-norm regularization strength, and turns on L_0-norm regularization, enforcing sparsity for Potts couplings (default: 1e-4)
- `-l0` turns on L_0-norm regularization, but **without** setting the regularization strength
- `-ss` specifies an input "secondary structure" file used to specify the initial set of clusters to consider in the expansion
- `-lax` enables a laxer cluster construction rule, increasing the number of clusters included in the cluster expansion routine

Additional QLS options

- `-c` specifies the set of (true) correlations to compare with for the MC learning routine (default: "input")
- `-e` sets the maximum tolerable error threshold; the program will run until all of the error terms $\epsilon_{p1}, \epsilon_{p2}, \epsilon_{\rm max} < e$ (default: 1)

Additional QGT options

- `-c` specifies the file giving the consensus sequence (default: "input")
- `-m` specifies the file containing the compressed representation of the data (i.e. the compressed MSA, default: "input")
- `-w` specifies the file with weights for each configuration in the data (default: "input")
- `-pthresh` sets the threshold for three-point correlations that will be printed (default: $10 \times \langle p \rangle$, where $\langle p \rangle$ is the average one-point correlation)
- `-p3` enables computation and comparison of three-point correlations; note that by default not all correlations are printed (see `-pthresh` option above)
- `-p3full` enables computation and comparison of three-point correlations and sets `pthresh` to zero
- `-nmax` set the maximum number of three-point correlations to print, another way

to control file size (default: none)
- `-msaout` enables output of sample configurations from the model and their energies

References

1. Barton, J. P., De Leonardis, E., Coucke, A. and Cocco, S. (2016). ACE: adaptive cluster expansion for maximum entropy graphical model inference. *Bioinformatics*, doi:http://dx.doi.org/10.1093/bioinformatics/btw328.
2. Cocco, S. and Monasson, R. (2011). Adaptive Cluster Expansion for Inferring Boltzmann Machines with Noisy Data. *Physical Review Letters*, **106**, 090601.
3. Cocco, S. and Monasson, R. (2012). Adaptive Cluster Expansion for the Inverse Ising Problem: Convergence, Algorithm and Tests. *Journal of Statistical Physics*, **147**(2), 252–314.
4. Barton, J. and Cocco, S. (2013). Ising models for neural activity inferred via selective cluster expansion: structural and coding properties. *Journal of Statistical Mechanics: Theory and Experiment*, **2013**(03), P03002.