EPAD Application Programming Interface (Version 2.3)

Feng Zhao and Jinbo Xu
Toyota Technological Institute at Chicago
1/20/2013

Introduction

Note:

1 EPAD currently is only tested with our Linux systems;

2 Please read the software manual at http://ttic.uchicago.edu/~jinbo/software/EPAD_guide.html before this EPAD API document;

3 EPAD uses epad.cfg in the config directory as its configuration file;

4 In epad.cfg file, the parameter MODEL_TYPE is set to TEMPLATE_BASED by default for the purpose of ranking models. You shall set MODEL_TYPE to FOLDING for the purpose of protein folding;

5 EPAD generates two intermediate feature files *.epad and *.epad_local in the SEQ/EPAD directory. See the above-mentioned EPAD_guide.html for feature generation.

The APIs calculating the potential of a residue pair or a protein model are included in the following files:

- **ScoringFunction.h and ScoringFunction.cpp**: source code for two classes ScoringByEPAD and ScoringByEPADLocal.
- **residue.h** (self contained): source code for two classes ATOM and RESIDUE.

The above API files use the following utility code: EPAD.h, EPADLocal.h, constraints.h, Score.h, strtokenizer.h, rmsdp.h, ScoreMatrix.h, sparselib.hh, EPAD.cpp, EPADLocal.cpp, rmsdp.cpp, constraints.cpp, and strtokenizer.cpp.

A small program testEPAD.cpp is provided to demonstrate how to use the EPAD APIs. This program reads in a set of protein decoys and then calculates their potentials. Since testEPAD uses the BALL (Biochemical Algorithms Library) library to parse a PDB file, BALL has to be installed to compile testEPAD.cpp. **However, the EPAD APIs do not depend on BALL, so you do not have to install BALL in order to use the EPAD APIs. You can also write your own test code without using BALL.**

A makefile is enclosed in the package to help you compile testEPAD. You may have to do some minor revisions of the makefile depending on your own Linux systems.

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ATOM
RESIDUE
ScoringFunction
    ScoringByEPAD

ScoringByEPADLocal

Class Documentation

ScoringByEPAD Class

This class provides interfaces to calculate the long range (chain distance >= 6) EPAD potential of a protein model. In the README, we assume the protein name is 1ektA.

- int **Init** (string config_file, string feature_file)

  This function reads in the configuration file epad.cfg and the *.epad file, initializes the potential function and generates the distance distribution of each residue pair. It returns 1 if initialization is successful and 0 otherwise.

  Example:

  ScoringFunction* sf=new ScoringByEPAD();

  sf->Init("./config/epad.cfg", "./SEQ/EPAD/1ektA.epad");

- double **evaluate** (vector< RESIDUE > &structure, int start, int end)

  This function takes a protein structure as input, and returns the EPAD potential between the residue *start* and residue *end*. The residue number starts from 0. If the input parameter *end* is -1, it indicates the last residue.

  Example:

  vector<RESIDUE> g_backbone; // initialized somewhere before, containing the 3D structure of a decoy

  double energy = sf->evaluate(g_backbone, 0, -1);

The documentation for this class originates from the following files:
ScoringFunction.h
ScoringFunction.cpp

ScoringByEPADLocal Class

This class provides interfaces to calculate the short range (chain distance < 6) EPAD potential of a protein structure. Here we assume the protein name is 1ektA.

- int **Init**(string modelFile, string feature_file, string labelFile)

  This function takes as input the model file modelCNFo1SS.dat in the config directory, the *.epad_local file, and another auxiliary label file cnf_label_SS_1.dat in the config directory. It initializes the EPADLocal potential function and generates the distance distribution of local residue pairs. It returns 1 if initialization is successful and 0 otherwise.

  Example:

  ScoringFunction* sf=new ScoringByEPADLocal();

  sf->Init("./config/modelCNFo1SS.dat",

  "./SEQ/EPAD/1ektA. epad_local", "./config/cnf_label_SS_1.dat");

- double **evaluate**(vector< RESIDUE > &structure, int start, int end)

  This function takes a peptide structure as input, and returns the EPADLocal potential between the residue *start* and residue *end*. Residues are numbered starting from 0. If the input parameter *end* is -1, it indicates the last residue.

  Example:

  vector<RESIDUE> g_backbone; // initialized somewhere before, containing the 3D structure of a decoy

  double energy = sf ->evaluate(g_backbone, 0, -1);

The documentation for this class originates from the following files:
ScoringFunction.h
ScoringFunction.cpp

ATOM Class

To calculate the potential of a structure, the user is expected to create a vector of RESIDUEs and pass it to the **evaluate**() function mentioned above. An ATOM object is the basic building block of a RESIDUE object.

- ATOM (string aname, int i)

  The constructor of the ATOM class, users need to provide the name of the ATOM, which we support 7 types including "CA", "CB", "O", "N", "C", "HN", and "SG". The 7 types are assigned a name_idx from 0 to 6 as in the order shown above. The second parameter required denotes the location in the sequence (residue index) of the residue containing this atom. The residue index is assumed to start from 0.

- string name

  Name of the atom.

- int name_idx

  Index of the name.

- double x, y, z

  3D-Coordinate of the atom.

- int res_idx

  Index of the residue containing this atom.

The documentation for this class was generated from the following file:
residue.h

RESIDUE Class

To calculate the potential of a structure, users need to create a vector of RESIDUEs and pass it to the **evaluate**() function mentioned above.

- void addAtom (ATOM atom)

This function adds an atom to the residue.

- void deleteAtom (string aname)

  This function removes an atom from the residue by the atom name.

- ATOM getAtom (string aname)

  This function retrieves an atom in the residue by the atom name.

- ATOM getAtom (int at)

  This function retrieves an atom in the residue by the atom name index. Please refer to the constructor of ATOM class for detailed definition of the name index.

- string name

  Name of the residue.

- int name_idx

  Name index of the residue. It is defined as followed: "ALA"=0, "ARG"=1, "ASN"=2, "ASP"=3, "CYS"=4, "GLN"=5, "GLU"=6, "GLY"=7, "HIS"=8, "ILE"=9, "LEU"=10, "LYS"=11, "MET"=12, "PHE"=13, "PRO"=14, "SER"=15, "THR"=16, "TRP"=17, "TYR"=18, "VAL"=19, "HET"=20.

- int seq

  The location of this residue in the sequence.

The documentation for this class originates from the following file:
residue.h