gift

2.0

Generated by Doxygen 1.8.12

Contents

1 Namespace Index **1**

1.1 Namespace List 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Namespace Documentation **7**

4.1 gift Namespace Reference 7

4.1.1 Typedef Documentation 8

4.1.1.1 IntArrayList 8

4.1.1.2 IntList 8

4.1.1.3 name2IndexHash 8

4.1.1.4 nameList 8

4.1.1.5 numericMatrix 8

4.1.2 Function Documentation 8

4.1.2.1 author(""Songpeng Zu"") 8

4.1.2.2 BoolToString(bool b) 8

4.1.2.3 email(""zusongpeng"@gmail.com"") 9

4.1.2.4 functionThread(func useFun, int thread, EM ∗point) 9

4.1.2.5 getIndexFromHash(const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIndexList, nameList &existNameList) 9

4.1.2.6 helpGift() 9

4.1.2.7 Matrix2Fingerpints(const std::string inputFile, IntArrayList &getFp, std::string delims) . . . 11
4.1.2.8 Matrix2FingerprintsByColumn(const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims) . . . 11
4.1.2.9 outRecord(parameters &EMparameters) . . . 12
4.1.2.10 printIntArrayList(const IntArrayList &fromIntArrayList) . . . 12
4.1.2.11 printMatrix(const numericMatrix &fromMatrix) . . . 13
4.1.2.12 readMatrix(const std::string inputFile, numericMatrix &getMat, std::string delims) 13
4.1.2.13 readName2IndexHash(const nameList fromNameList, name2IndexHash &name2Index) . . . 14
4.1.2.14 readNameListFromFile(const std::string inputFile, nameList &tonameList) . . . 14
4.1.2.15 readNameMatrixFromFile(const std::string inputFile, nameList &tonameList, Int←ArrayList &getFP, std::string delims) . . . 14
4.1.2.16 rowColFile(const std::string inputFile, rowCol &matrixRec, std::string delims) . . 15
4.1.2.17 updateTime(""2016-03-06"") . . . 16
4.1.2.18 version(""gift-2.0"") . . . 16
4.1.2.19 writeMatrix(const std::string outFileName, numericMatrix &resultMat, std::string delims) . . . 16
4.1.3 Variable Documentation . . . 16
4.1.3.1 domain2proteinList . . . 16
4.1.3.2 drug2proteinList . . . 16
4.1.3.3 drug2subList . . . 16
4.1.3.4 drugName2Index . . . 16
4.1.3.5 drugNameList . . . 16
4.1.3.6 drugSub2proteinSubMatrix . . . 17
4.1.3.7 drugSubNameList . . . 17
4.1.3.8 loglikelyArray . . . 17
4.1.3.9 observedDrug2ProteinMatrix . . . 17
4.1.3.10 predictDrug2SubList . . . 17
4.1.3.11 predictDrugNameList . . . 17
4.1.3.12 predictDrugNameList_WithSubs . . . 17
4.1.3.13 predictProtein2SubList . . . 17
4.1.3.14 predictProteinNameList . . . 17
4.1.3.15 predictProteinNameList_WithSubs . . . 17
4.1.3.16 protein2domainList . . . 18
4.1.3.17 proteinName2Index . . . 18
4.1.3.18 proteinNameList . . . 18
4.1.3.19 proteinSubNameList . . . 18
4.1.3.20 recLogLeastNum . . . 18
4.1.3.21 sub2drugList . . . 18
4.1.3.22 vardrugSub2proteinSubMatrix . . . 18

5 Class Documentation 19

5.1 gift::EM Class Reference 19

5.1.1 Constructor & Destructor Documentation 20

5.1.1.1 EM(parameters &param) 20

5.1.1.2 ∼EM() 20

5.1.2 Member Function Documentation 20

5.1.2.1 EStep() 20

5.1.2.2 EStep(int) 20

5.1.2.3 EStepThread(int threadNth) 21

5.1.2.4 functionThread(void(EM::∗function)(int), int thread) 21

5.1.2.5 iterdrugSub2ProteinSub(int drugIndex, int proteinIndex) 21

5.1.2.6 MStep() 21

5.1.2.7 MStep(int) 22

5.1.2.8 MStepThread(int threadNth) 22

5.1.2.9 outTrainResult() 22

5.1.2.10 outTrainVariance() 23

5.1.2.11 predictDrugs() 23

5.1.2.12 predictDrugsProteins() 24

5.1.2.13 predictDrugsProteinsWithSubs() 24

5.1.2.14 predictDrugsWithSubs() 25

5.1.2.15 predictDrugsWithSubsProteins() 26

5.1.2.16 predictDrugsWithSubsProteinsWithSubs() 26

5.1.2.17 predictEM() 27

5.1.2.18 predictProteins() 28

5.1.2.19 predictProteinsWithSubs() 28

5.1.2.20 recLoglikely() 29

5.1.2.21 setLoglikely(double logscore) 29

5.1.2.22 setPointerDrug2Protein(IntArrayList &d2p) 29

5.1.2.23 setPointerDrug2Sub(IntArrayList &d2s) 29

5.1.2.24 setPointerDrugSub2ProteinSub(numericMatrix &ds2ps) 30

5.1.2.25 setPointerProtein2Sub(IntArrayList &p2s) 30
5.1.2.26 trainEM() 30
5.1.2.27 varEM() 31
5.2 gift::parameters Class Reference 31
5.2.1 Constructor & Destructor Documentation 32
5.2.1.1 parameters(const std::string) 32
5.2.2 Member Function Documentation 34
5.2.2.1 InitDrugName2Index() 34
5.2.2.2 InitDrugSub2ProteinSub() 35
5.2.2.3 InitDrugSubNameList() 35
5.2.2.4 InitObservedDrug2ProteinMatrix() 36
5.2.2.5 InitPredictParameters() 36
5.2.2.6 InitProteinName2Index() 36
5.2.2.7 InitProteinSubNameList() 37
5.2.2.8 InitVarDrugSub2ProteinSub() 37
5.2.2.9 setDomainNum(int number) 37
5.2.2.10 setDrugNum(int number) 37
5.2.2.11 setProteinNum(int number) 37
5.2.2.12 setSubNum(int number) 37
5.2.3 Member Data Documentation 37
5.2.3.1 alphaEB 37
5.2.3.2 betaEB 38
5.2.3.3 chemfpRec 38
5.2.3.4 CPIsRec 38
5.2.3.5 domainNum 38
5.2.3.6 drug2proteinFileName 38
5.2.3.7 drug2subFileName 38
5.2.3.8 drugNameListFile 38
5.2.3.9 drugNum 38
5.2.3.10 drugSub2proteinSubFileName 38

5.2.3.11 drugSubNameListFile 38
5.2.3.12 fn 38
5.2.3.13 fp 38
5.2.3.14 inputDelims 38
5.2.3.15 iterNum 39
5.2.3.16 loglikelyRecord 39
5.2.3.17 outDrugSub2ProteinSubFileName 39
5.2.3.18 outPredictCPIsFileName 39
5.2.3.19 outputDelims 39
5.2.3.20 outRecordFileName 39
5.2.3.21 outVarDrugSub2proteinSubFileName 39
5.2.3.22 predictDrugsFileName 39
5.2.3.23 predictDrugsFileName_WithSubs 39
5.2.3.24 predictProteinsFileName 39
5.2.3.25 predictProteinsFileName_WithSubs 39
5.2.3.26 protein2subFileName 39
5.2.3.27 proteinfpRec 39
5.2.3.28 proteinNameListFile 39
5.2.3.29 proteinNum 40
5.2.3.30 proteinSubNameListFile 40
5.2.3.31 subNum 40
5.2.3.32 task 40
5.2.3.33 thread 40
5.3 gift::rowCol Class Reference 40
5.3.1 Constructor & Destructor Documentation 40
5.3.1.1 rowCol(int row, int col) 40
5.3.1.2 rowCol() 41
5.3.2 Member Data Documentation 41
5.3.2.1 colNum 41
5.3.2.2 rowNum 41

6 File Documentation 43

6.1 EMGift.cpp File Reference 43
6.2 functionsGift.cpp File Reference 43
6.3 gift.hpp File Reference 44
6.4 main.cpp File Reference 46
6.4.1 Function Documentation 46
6.4.1.1 main(int argc, char **argv) 46
6.5 parameterGift.cpp File Reference 47

Index 49

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

gift 7

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

gift::EM 19
gift::parameters 31
gift::rowCol 40

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

EMGift.cpp 43
functionsGift.cpp 43
gift.hpp 44
main.cpp 46
parameterGift.cpp 47

Chapter 4

Namespace Documentation

4.1 gift Namespace Reference

Classes

- class EM
- class parameters
- class rowCol

Typedefs

- typedef std::vector< int > IntList
- typedef std::vector< std::vector< int > > IntArrayList
- typedef std::vector< std::vector< double > > numericMatrix
- typedef std::map< std::string, int > name2IndexHash
- typedef std::vector< std::string > nameList

Functions

- int Matrix2Fingerpints (const std::string inputFile, IntArrayList &getFp, std::string delims)
- int readMatrix (const std::string inputFile, numericMatrix &getMat, std::string delims)
- int rowColFile (const std::string inputFile, rowCol &matrixRec, std::string delims)
- int writeMatrix (const std::string outFileName, numericMatrix &resultMat, std::string delims)
- int printIntArrayList (const IntArrayList &fromIntArrayList)
- int printMatrix (const numericMatrix &fromMatrix)
- int readNameListFromFile (const std::string inputFile, nameList &tonameList)
- int readNameMatrixFromFile (const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std↩
  ::string delims)
- int readName2IndexHash (const nameList fromNameList, name2IndexHash &name2Index)
- int getIndexFromHash (const name2IndexHash &name2Index, const nameList fromNameList, IntList &to↩
  IndexList, nameList &existNameList)
- int helpGift ()
- int outRecord (parameters &EMparameters)
- int Matrix2FingerprintsByColumn (const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)
- const std::string author ("Songpeng Zu")
- const std::string email ("zusongpeng@gmail.com")
- const std::string version ("gift-2.0")
- const std::string updateTime ("2016-03-06")
- template<typename func >
  int functionThread (func useFun, int thread, EM ∗point)
- const char ∗ BoolToString (bool b)

Variables

- const int recLogLeastNum = 5
- IntArrayList drug2proteinList
- IntArrayList drug2subList
- IntArrayList sub2drugList
- IntArrayList protein2domainList
- IntArrayList domain2proteinList
- numericMatrix drugSub2proteinSubMatrix
- numericMatrix observedDrug2ProteinMatrix
- numericMatrix vardrugSub2proteinSubMatrix
- std::vector< double > loglikelyArray
- name2IndexHash drugName2Index
- name2IndexHash proteinName2Index
- nameList drugNameList
- nameList proteinNameList
- nameList drugSubNameList
- nameList proteinSubNameList
- nameList predictDrugNameList
- nameList predictProteinNameList
- nameList predictDrugNameList_WithSubs
- nameList predictProteinNameList_WithSubs
- IntArrayList predictDrug2SubList
- IntArrayList predictProtein2SubList

4.1.1 Typedef Documentation

4.1.1.1 typedef std::vector<std::vector<int> > **gift::IntArrayList**

4.1.1.2 typedef std::vector<int> **gift::IntList**

4.1.1.3 typedef std::map<std::string,int> **gift::name2IndexHash**

4.1.1.4 typedef std::vector<std::string> **gift::nameList**

4.1.1.5 typedef std::vector<std::vector<double> > **gift::numericMatrix**

4.1.2 Function Documentation

4.1.2.1 const std::string gift::author ("Songpeng Zu")

Referenced by helpGift(), and outRecord().

4.1.2.2 const char∗ gift::BoolToString (bool *b*) [inline]

Referenced by gift::parameters::parameters().

```
                                             {
      return b ? "true" : "false";
   } // end of function BoolToString
```

4.1.2.3 const std::string gift::email ("zusongpeng@gmail.com")

Referenced by helpGift(), and outRecord().

4.1.2.4 template<typename func > int gift::functionThread (func *useFun,* int *thread,* **EM** ∗ *point*)

Referenced by gift::EM::EStep(), and gift::EM::MStep().

```
                                                        {
    boost::thread * y;
    boost::thread_group * x = new boost::thread_group;
    for(int i=0;i<thread;++i){
      y = new boost::thread(useFun,point,i);
      x->add_thread(y);
    } // end of loop i
    x->join_all();
    delete x;
    return 0;
  } // end of function.
```

4.1.2.5 int gift::getIndexFromHash (const **name2IndexHash** & *name2Index,* const **nameList** *fromNameList,* **IntList** & *toIndexList,* **nameList** & *existNameList*)

Referenced by gift::EM::predictDrugs(), gift::EM::predictDrugsProteins(), gift::EM::predictDrugsProteinsWithSubs(), gift::EM::predictDrugsWithSubsProteins(), and gift::EM::predictProteins().

```
                                               {
    for(const auto fromName : fromNameList){
      if (name2Index.find(fromName) != name2Index.end()){
        existNameList.push_back(fromName);
        toIndexList.push_back( (name2Index.find(fromName))->second);
      } else {
        std::cout<<"Cannot find the key " << fromName
                 <<" fromNameList. Continue..." <<std::endl;
      } // end of if else
    } // end of loop fromNameList
    return 0;
  } // end of function
```

4.1.2.6 int gift::helpGift ( )

References author(), email(), updateTime(), and version().

Referenced by main().

```
                  {
    // Output gift information and useness to standard output.
    // Basic information about gift.
    std::cout<<"Gift is used to predict compound-protein interactions based on "
             <<std::endl;
    std::cout<<"their substructures interactions."<<std::endl;
    std::cout<<"It is also used to infer the substructres interactions from"
             <<std::endl;
    std::cout<<" the known drug-protein interactions."<<std::endl;
    std::cout<<"If you want to know more about gift, please read the paper: "
             <<std::endl;
    std::cout<<"Global Optimization-based Inference of Chemogenomic Features "
             <<std::endl;
    std::cout<<"from Drug-Target Interactions, which is published  "<<std::endl;
    std::cout<<"on Bioinformatics, 2015. "<<std::endl;
    std::cout<<"Author: "<<author<<std::endl;
    std::cout<<"Email: " <<email<<std::endl;
    std::cout<<"Current version: "<<version<<std::endl;
    std::cout<<"Last update time: "<<updateTime<<std::endl;
    std::cout<<"You can get the C++ source code from: "<<std::endl;
    std::cout<<"https://github.com/songpeng/GIFT" << std::endl;
    std::cout<<std::endl;

    //Input parameters
    std::cout<<"--help | -h to show the help information of gift."<<std::endl;
    std::cout<<"--version | -v to show the version information of gift."
             <<std::endl;
    std::cout<<"Gift need one configure file for its running."<<std::endl;
    std::cout<<"Please use --config to tell gift the configure file name."
             <<std::endl;
    std::cout<<"The content in the configure file are listed below: "<<std::endl;

    // configure file information.
    std::cout<<"[INPUT DATA FILE NAMES]" <<std::endl;
    std::cout<<"drug2proteinFileName=<string> : "
             <<"file name for drug protein interactions" <<std::endl;
    std::cout<<"drug2subFilename=<string> : "
             <<"file name for drug to substructure" <<std::endl;
    std::cout<<"protein2subFileName=<string> : "
             <<"file name for protein to substructure" << std::endl;
    std::cout<<"drugSub2proteinSubfilename=<string> : "
             <<"file name for drugSub to proteinSub interaction probability."
             <<std::endl;
    std::cout<<"drugNameListFile=<string> : "
             <<"file name for drug names" << std::endl;
    std::cout<<"drugSubNameListFile=<string> : "
             <<"file name for drug substructures names." << std::endl;
    std::cout<<"proteinNameListFile=<string> : "
             <<"file name for protein names" << std::endl;
    std::cout<<"proteinSubNameListFile=<string> : "
             <<"file name for protein substructures names." << std::endl;

    std::cout<<"[INPUT PARAMETERS FOR EM ALGORITHM]" <<std::endl;
    std::cout<<"alphaEB=<double> : "
             <<"parameter for Empricial Bayesian estimates for initEM."
             <<std::endl;
    std::cout<<"betaEB=<double> : "
             <<"parameter for Empricial Bayesian estimates for initEM."
             <<std::endl;
    std::cout<<"fp=<double> : "<<"false positive rate"<<std::endl;
    std::cout<<"fn=<double> : "<<"false negative rate"<<std::endl;
    std::cout<<"threadNum=<int> : "<<"thread number for EM." <<std::endl;
    std::cout<<"EMIterationNum=<int> : "<<"iteration numbers/steps for EM."<<std::endl;
    std::cout<<"task=<string> : "<<"run gift for [train] or [predict]."<<std::endl;
    std::cout<<"loglikelyRecord=<string> : "<<
      "record [true] or not [false] the loglikely in in every step."<<std::endl;
    std::cout<<"inputDelims=<string> : "
             <<"sep character for input files, such as  '\t',',' "<<std::endl;

    std::cout<<"[INPUT FILE VERSION INFORMATION]"<<std::endl;
    std::cout<<"chemFingerPrintRecord=<string> : "
             <<"source and version of chemical fingerprints."<<std::endl;
    std::cout<<"proteinFingerPrintRecord=<string> : "
             <<"source and version of protein fingerpints/domains."<<std::endl;
    std::cout<<"comProteinInteractionRecord=<string> : "
             <<"source and version of compound-protein interactions."<<std::endl;

    std::cout<<"[INPUT FILE NAME FOR PREDICTION]"<<std::endl;
    std::cout<<"predictDrugsFileName=<string> : "
             <<"file name for drug names used for prediction by gift."<<std::endl;
    std::cout<<"predictProteinsFileName=<string> : "
             <<"file name for protein names used for prediction by gift."<<std::endl;
    std::cout<<"predictDrugsFileName_WithSubs=<string> : "
             <<"file name for drug names together with their substructures."
             <<std::endl;
    std::cout<<"predictProteinsFileName_WithSubs=<string> : "
             <<"file name for protein names together with their substructures."
             <<std::endl;

    std::cout<<"[OUTPUT FILE NAME AND FORMAT]"<<std::endl;
    std::cout<<"outputDelims=<string> : "
             <<"sep character for output files." <<std::endl;
    std::cout<<"outRecordFileName=<string> : "<<"file name for output records."
             <<std::endl;
    std::cout<<"outPredictCPIsFileName=<string> : "
             <<"file name for output CPIs." <<std::endl;
    std::cout<<"outDrugSub2ProteinSubFileName=<string> : "
             <<"file name for output drugSub2proteinSub matrix."<<std::endl;
    std::cout<<"outVarDrugSub2proteinSubFileName=<string> : "
             <<"file name for output variance of drugSub2proteinSub." <<std::endl;

    return 0;
  } // end of function
```

4.1.2.7 int gift::Matrix2Fingerpints (const std::string *inputFile,* IntArrayList & *getFp,* std::string *delims*)

Referenced by gift::parameters::parameters().

```
                                          {
    std::ifstream input (inputFile, std::ios::in);
    std::string line;
    std::vector<std::string> array;
    std::vector<int> tempRec;
    while (std::getline(input,line)) {
      boost::algorithm::split(array,line,boost::is_any_of(delims));
      int arraylen = array.size();
      for (int i=0;i<arraylen;++i) {
        if (array[i].compare("1") == 0) {
          tempRec.push_back(i);
        }// end of if
      } // end of for
      getFp.push_back(tempRec);
      tempRec.clear();
    } // end of while

    // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    // try {
    //   input.open(inputFile, std::ifstream::in);
    //   if (input.peek() == std::ifstream::traits_type::eof()){
    //     std::cerr <<inputFile <<" is empty. " <<std::endl;
    //     return 1;
    //   } // end of if
    //   while (std::getline(input,line)) {
    //     boost::algorithm::split(array,line,boost::is_any_of(delims));
    //     int arraylen = array.size();
    //     for (int i=0;i<arraylen;++i) {
    //       if (array[i].compare("1") == 0) {
    //         tempRec.push_back(i);
    //       }// end of if
    //     } // end of for
    //     getFp.push_back(tempRec);
    //     tempRec.clear();
    //   } // end of while
    // } catch (std::ifstream::failure e) {
    //   std::cerr <<"Exceptions open/read file "<<inputFile<<std::endl;
    //   return 1;
    // } // end of catch

    return 0;
  } // end of function.
```

4.1.2.8 int gift::Matrix2FingerprintsByColumn (const std::string *inputFile,* IntArrayList & *getFP,* int *rowNum,* std::string *delims*)

Referenced by gift::parameters::parameters().

```
                                                  {
    std::ifstream input (inputFile, std::ios::in);
    std::string line;
    std::vector<std::string> array;
    int linenum = 0;
    // init getFP first.
    std::cout<<"Init getFP IntArrayList..."<<std::endl;
    std::vector<int> tmpArray;
    for(int i=0;i<rowNum;++i) {
      getFP.push_back(tmpArray);
    } // end of loop
    std::cout<<"End of Init getFP IntArrayList."<<std::endl;

    while (std::getline(input,line)){
      boost::algorithm::split(array,line,boost::is_any_of(delims));
      int arraylen = array.size();
      for (int i=0;i<arraylen;++i){
        if (array[i].compare("1") == 0){
          getFP[i].push_back(linenum);
        } // end of if
      } // end of loop for i
      linenum += 1;
      array.clear();
    } // end of while for file read.

    // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    // try {
    //   input.open(inputFile,std::ifstream::in);
    //   if(input.peek() == std::ifstream::traits_type::eof()){
    //     std::cerr<< inputFile << " is empty. "<<std::endl;
    //     return 1;
    //   } // end of if

    //   // init getFP first.
    //   for(int i=0;i<rowNum;++i) {
    //     std::vector<int> tmpArray;
    //     getFP.push_back(tmpArray);
    //   } // end of loop

    //   // read file.
    //   while (std::getline(input,line)){
    //     boost::algorithm::split(array,line,boost::is_any_of(delims));
    //     int arraylen = array.size();
    //     for (int i=0;i<arraylen;++i){
    //       if (array[i].compare("1") == 0){
    //         getFP[i].push_back(linenum);
    //       } // end of if
    //     } // end of loop for i
    //     linenum += 1;
    //   } // end of while for file read.
    // } catch (std::ifstream::failure e) {
    //   std::cerr<<"Exceptions open/read file "<<inputFile<<std::endl;
    //   return 1;
    // } // end of catch
    return 0;
  } // end of function
```

4.1.2.9 int gift::outRecord (parameters & *EMparameters*)

References author(), email(), loglikelyArray, gift::parameters::loglikelyRecord, gift::parameters::outRecordFileName, gift::parameters::task, updateTime(), and version().

Referenced by main().

```
                                          {
    std::ofstream output (EMparameters.outRecordFileName,std::ofstream::out);
    if (!output.is_open()){
      std::cerr<<"Error open file "<<EMparameters.outRecordFileName<<std::endl;
      return 1;
    }// end of if
    // Basic information about gift.
    output<<"The author of gift is  " << author <<std::endl;
    output<<"Contact information: " << email <<std::endl;
    output<<"Current gift's version is "<< version <<std::endl;
    output<<"Update time is " << updateTime <<std::endl;
    // Running information.
    std::chrono::system_clock::time_point timePos =
      std::chrono::system_clock::now();
    std::time_t timePosT = std::chrono::system_clock::to_time_t(timePos);
    output<<"The job destination is "<<EMparameters.task;
    output<<", which is finished at "<<std::ctime(&timePosT) <<std::endl;
    if(EMparameters.task.compare("train") == 0){
      if(EMparameters.loglikelyRecord){
        output<<"The loglikelyhood values are followed: "<<std::endl;
        for(const auto m : loglikelyArray){
          output<<m<<std::endl;
        } // end of loop for m.
      } // end of if
    } // end of if
    output.close();
    return 0;
  } // end of function
```

4.1.2.10 int gift::printIntArrayList (const IntArrayList & *fromIntArrayList*)

Referenced by gift::parameters::parameters().

```
                                                            {
    int lineNum = fromIntArrayList.size();
    for(int i=0;i<lineNum;++i){
      if (fromIntArrayList[i].empty()){
        std::cerr<<"[ERROR]: Row "<<i<<" is empty for printing... "<<std::endl;
        return 1;
      } // end of if
      for(const auto m : fromIntArrayList[i]){std::cout<<m<<",";}
      std::cout<<std::endl;
    } // end of loop for i
    return 0;
  } // end of function.
```

4.1.2.11 int gift::printMatrix (const numericMatrix & *fromMatrix*)

```
                                                  {
    int rowNum = fromMatrix.size();
    int colNum = fromMatrix[0].size();
    for(int i=0;i<rowNum;++i){
      for(int j=0;j<colNum;++j){
        std::cout<<fromMatrix[i][j]<<",";
      } // end of loop for j
      std::cout<<std::endl;
    } // end of loop for i
    return 0;
  } // end of function
```

4.1.2.12 int gift::readMatrix (const std::string *inputFile,* numericMatrix & *getMat,* std::string *delims*)

Referenced by gift::parameters::InitDrugSub2ProteinSub().

```
                                  {
    std::ifstream input (inputFile, std::ios::in);
    std::string line;
    std::vector<std::string> array;
    std::vector<double> tempRec;
    while(std::getline(input,line)){
      boost::algorithm::split(array,line,boost::is_any_of(delims));
      int arraylen = array.size();
      for(int i=0;i<arraylen;++i){
        std::string::size_type* idx = 0;
        tempRec.push_back(std::stod(array[i], idx));
      } // end of for
      getMat.push_back(tempRec);
      tempRec.clear();
    } // end of while

    // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    // try {
    //   input.open(inputFile, std::ifstream::in);
    //   if (input.peek() == std::ifstream::traits_type::eof()){
    //     std::cerr <<inputFile <<" is empty. " <<std::endl;
    //   } // end of if
    //   while(std::getline(input,line)){
    //     boost::algorithm::split(array,line,boost::is_any_of(delims));
    //     int arraylen = array.size();
    //     for(int i=0;i<arraylen;++i){
    //       std::string::size_type* idx = 0;
    //       tempRec.push_back(std::stod(array[i], idx));
    //     } // end of for
    //     getMat.push_back(tempRec);
    //     tempRec.clear();
    //   } // end of while
    // } catch (std::ifstream::failure e) {
    //   std::cerr <<"Exceptions open/read file "<<inputFile<<std::endl;
    //   return 1;
    // } // end of catch
    return 0;
  }// end of function.
```

4.1.2.13 int gift::readName2IndexHash (const nameList *fromNameList*, name2IndexHash & *name2Index*)

Referenced by gift::parameters::InitDrugName2Index(), and gift::parameters::InitProteinName2Index().

```
                                                  {
      // fromNameList should be in order.
      if (fromNameList.empty()){
        std::cerr<<"The fromNameList is empty. "<<std::endl;
        return 1;
      } // end of if
      int recordIndex = 0;
      for(const auto fromName : fromNameList){
        name2Index.insert(std::pair<std::string,int>(fromName,recordIndex));
        ++recordIndex;
      } // end of loop fromNameList
      return 0;
    } // end of function
```

4.1.2.14 int gift::readNameListFromFile (const std::string *inputFile*, nameList & *tonameList*)

Referenced by gift::parameters::InitDrugName2Index(), gift::parameters::InitDrugSubNameList(), gift::parameters::InitPredictParameters(), gift::parameters::InitProteinName2Index(), and gift::parameters::InitProteinSubNameList().

```
                                                                         {
      // each line in the file represents one name.
      // line should end with "\n", not "[\r\t]\n"
      std::ifstream input (inputFile, std::ios::in);
      std::string line;
      while (std::getline(input,line)) {
        tonameList.push_back(line);
      } // end of while
      input.close();

      // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
      // try {
      //   input.open(inputFile,std::ifstream::in);
      //   if (input.peek() == std::ifstream::traits_type::eof() ){
      //     std::cerr << inputFile <<" is empty. "<<std::endl;
      //     return 1;
      //   } // end of if
      //   while (std::getline(input,line)) {
      //     tonameList.push_back(line);
      //   } // end of while
      //   input.close();
      // } catch (std::ifstream::failure e) {
      //   std::cerr<<"Exceptions open/read file " << inputFile<<std::endl;
      // } // end of try catch
      return 0;
    } // end of function
```

4.1.2.15 int gift::readNameMatrixFromFile (const std::string *inputFile*, nameList & *tonameList*, IntArrayList & *getFP*, std::string *delims*)

Referenced by gift::parameters::InitPredictParameters().

```
                                                                 {
      std::ifstream input (inputFile, std::ios::in);
      std::string line;
      std::vector<std::string> array;
      std::vector<int> tempRec;
      while(std::getline(input,line)) {
        boost::algorithm::split(array,line,boost::is_any_of(delims));
        tonameList.push_back(array[0]); // first column is name.
        int arraylen = array.size();
        for(int i=1;i<arraylen;++i){
          if(array[i].compare("1") == 0) {
            tempRec.push_back(i-1); // Use i-1, since first column is name.
          } // end of if
        } // end of loop for i.
        getFP.push_back(tempRec);
        tempRec.clear();
      } // end of while

      // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
      // try {
      //   input.open(inputFile, std::ifstream::in);
      //   if (input.peek() == std::ifstream::traits_type::eof()) {
      //     std::cerr<<inputFile<<" is empty. "<<std::endl;
      //     return 1;
      //   } // end of if
      //   while(std::getline(input,line)) {
      //     boost::algorithm::split(array,line,boost::is_any_of(delims));
      //     tonameList.push_back(array[0]); // first column is name.
      //     int arraylen = array.size();
      //     for(int i=1;i<arraylen;++i){
      //       if(array[i].compare("1") == 0) {
      //         tempRec.push_back(i-1); // Use i-1, since first column is name.
      //       } // end of if
      //     } // end of loop for i.
      //     getFP.push_back(tempRec);
      //     tempRec.clear();
      //   } // end of while
      // } catch (std::ifstream::failure e) {
      //   std::cerr<<"Exceptions open/read file "<<inputFile<<std::endl;
      //   return 1;
      // } // end of catch
      return 0;
    } // end of function
```

4.1.2.16 int gift::rowColFile (const std::string *inputFile*, rowCol & *matrixRec*, std::string *delims*)

References gift::rowCol::colNum, and gift::rowCol::rowNum.

Referenced by gift::parameters::parameters().

```
                                 {
    std::ifstream input (inputFile, std::ios::in);
    std::string line;
    int count = 0;
    std::vector<std::string> array;

    std::getline(input,line);
    ++count;
    boost::algorithm::split(array, line, boost::is_any_of(delims));
    matrixRec.colNum = array.size();
    // string getline func over istream.
    while(std::getline(input,line)){
      // QUESTION: how about empty line?
      ++count;
    } // end of while
    input.close();

    matrixRec.rowNum = count;
    matrixRec.colNum = array.size();

    // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
    // try {
    //   input.open(inputFile,std::ifstream::in);
    //   if (input.peek() == std::ifstream::traits_type::eof()){
    //     std::cerr <<inputFile<<" is empty."<<std::endl;
    //     return 1;
    //   }

    //   std::getline(input,line);
    //   ++count;
    //   boost::algorithm::split(array, line, boost::is_any_of(delims));
    //   matrixRec.colNum = array.size();
    //   // string getline func over istream.
    //   while(std::getline(input,line)){
    //     // QUESTION: how about empty line?
    //     ++count;
    //   } // end of while
    //   input.close();

    //   matrixRec.rowNum = count;
    //   matrixRec.colNum = array.size();
    // } catch (std::ifstream::failure e) {
    //   std::cerr << "Exceptions open/read file "<<inputFile<<std::endl;
    //   return 1;
    // } // end of catch
    return 0;
  } // end of function.
```

4.1.2.17 const std::string gift::updateTime ("2016-03-06")

Referenced by helpGift(), main(), and outRecord().

4.1.2.18 const std::string gift::version ("gift-2.0")

Referenced by helpGift(), main(), and outRecord().

4.1.2.19 int gift::writeMatrix (const std::string *outFileName,* numericMatrix & *resultMat,* std::string *delims*)

Referenced by gift::EM::outTrainResult(), and gift::EM::outTrainVariance().

```
                                    {
    std::ofstream output (outFileName,std::ios::out);
    if (output.is_open()) {
      using boost::algorithm::join;
      using boost::adaptors::transformed;
      for (numericMatrix::iterator it = resultMat.begin();
           it != resultMat.end(); ++it) {
        output << join(*it | transformed(static_cast<std::string(*)(double)>
                                          (std::to_string) ), delims);
        output<<std::endl;
      }// end of for
      output.close();
    } else {
      std::cerr<< "Error opening file " <<outFileName<<std::endl;
      return 1;
    } // end of if else
    return 0;
  }// end of function
```

4.1.3 Variable Documentation

4.1.3.1 IntArrayList gift::domain2proteinList

4.1.3.2 IntArrayList gift::drug2proteinList

4.1.3.3 IntArrayList gift::drug2subList

4.1.3.4 name2IndexHash gift::drugName2Index

Referenced by gift::EM::predictDrugs(), gift::EM::predictDrugsProteins(), and gift::EM::predictDrugsProteinsWith↩ Subs().

4.1.3.5 nameList gift::drugNameList

Referenced by gift::EM::predictProteins(), and gift::EM::predictProteinsWithSubs().

4.1.3.6 numericMatrix gift::drugSub2proteinSubMatrix

Referenced by gift::EM::MStep().

4.1.3.7 nameList gift::drugSubNameList

4.1.3.8 std::vector< double > gift::loglikelyArray

Referenced by outRecord().

4.1.3.9 numericMatrix gift::observedDrug2ProteinMatrix

Referenced by gift::EM::EStep().

4.1.3.10 IntArrayList gift::predictDrug2SubList

Referenced by gift::EM::predictDrugsWithSubs(), gift::EM::predictDrugsWithSubsProteins(), and gift::EM::predict↩
DrugsWithSubsProteinsWithSubs().

4.1.3.11 nameList gift::predictDrugNameList

Referenced by gift::EM::predictDrugs(), gift::EM::predictDrugsProteins(), gift::EM::predictDrugsProteinsWithSubs(), and gift::EM::predictEM().

4.1.3.12 nameList gift::predictDrugNameList_WithSubs

Referenced by gift::EM::predictDrugsWithSubs(), gift::EM::predictDrugsWithSubsProteins(), gift::EM::predict↩
DrugsWithSubsProteinsWithSubs(), and gift::EM::predictEM().

4.1.3.13 IntArrayList gift::predictProtein2SubList

Referenced by gift::EM::predictDrugsProteinsWithSubs(), gift::EM::predictDrugsWithSubsProteinsWithSubs(), and gift::EM::predictProteinsWithSubs().

4.1.3.14 nameList gift::predictProteinNameList

Referenced by gift::EM::predictDrugsProteins(), gift::EM::predictDrugsWithSubsProteins(), gift::EM::predictEM(), and gift::EM::predictProteins().

4.1.3.15 nameList gift::predictProteinNameList_WithSubs

Referenced by gift::EM::predictDrugsProteinsWithSubs(), gift::EM::predictDrugsWithSubsProteinsWithSubs(), gift::EM::predictEM(), and gift::EM::predictProteinsWithSubs().

4.1.3.16 IntArrayList gift::protein2domainList

4.1.3.17 name2IndexHash gift::proteinName2Index

Referenced by gift::EM::predictDrugsProteins(), gift::EM::predictDrugsWithSubsProteins(), and gift::EM::predict↩ Proteins().

4.1.3.18 nameList gift::proteinNameList

Referenced by gift::EM::predictDrugs(), and gift::EM::predictDrugsWithSubs().

4.1.3.19 nameList gift::proteinSubNameList

4.1.3.20 const int gift::recLogLeastNum = 5

Referenced by gift::EM::trainEM().

4.1.3.21 IntArrayList gift::sub2drugList

4.1.3.22 numericMatrix gift::vardrugSub2proteinSubMatrix

Chapter 5

Class Documentation

5.1 gift::EM Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- EM (parameters &param)
- ∼EM ()
- int setPointerDrug2Sub (IntArrayList &d2s)
- int setPointerProtein2Sub (IntArrayList &p2s)
- int setPointerDrug2Protein (IntArrayList &d2p)
- int setPointerDrugSub2ProteinSub (numericMatrix &ds2ps)
- double iterdrugSub2ProteinSub (int drugIndex, int proteinIndex)
- int functionThread (void(EM::∗function)(int), int thread)
- void EStepThread (int threadNth)
- int EStep ()
- int EStep (int)
- void MStepThread (int threadNth)
- int MStep ()
- int MStep (int)
- double recLoglikely ()
- int setLoglikely (double logscore)
- int trainEM ()
- int predictDrugs ()
- int predictProteins ()
- int predictDrugsWithSubs ()
- int predictProteinsWithSubs ()
- int predictDrugsWithSubsProteinsWithSubs ()
- int predictDrugsProteinsWithSubs ()
- int predictDrugsWithSubsProteins ()
- int predictDrugsProteins ()
- int predictEM ()
- int varEM ()
- int outTrainResult ()
- int outTrainVariance ()

5.1.1 Constructor & Destructor Documentation

5.1.1.1 gift::EM::EM (parameters & *param*) [inline]

```
        : loglikelyRecord(param.loglikelyRecord)
        , fn(param.fn)
        , fp(param.fp)
        , thread(param.thread)
        , iterNum(param.iterNum)
        , drugNum(param.drugNum)
        , subNum(param.subNum)
        , domainNum(param.domainNum)
        , proteinNum(param.proteinNum)
        , task(param.task)
        , drug2sub(&drug2subList)
        , sub2drug(&sub2drugList)
        , protein2sub(&protein2domainList)
        , sub2protein(&domain2proteinList)
        , drug2protein(&drug2proteinList)
        , drugSub2proteinSub(&drugSub2proteinSubMatrix)
        , observedDrug2Protein(&observedDrug2ProteinMatrix)
        , vardrugSub2proteinSub(&vardrugSub2proteinSubMatrix)
        , loglikely(&loglikelyArray)
        , predictDrugsFileName(param.predictDrugsFileName)
        , predictProteinsFileName(param.predictProteinsFileName)
        , predictDrugsFileName_WithSubs(param.predictDrugsFileName_WithSubs)
        , predictProteinsFileName_WithSubs(param.predictProteinsFileName_WithSubs)
        , outputDelims(param.outputDelims)
        , outRecordFileName(param.outRecordFileName)
        , outPredictCPIsFileName(param.outPredictCPIsFileName)
        , outDrugSub2ProteinSubFileName(param.outDrugSub2ProteinSubFileName)
        , outVarDrugSub2proteinSubFileName(param.outVarDrugSub2proteinSubFileName)
      { } // end of constuctor.
```

5.1.1.2 gift::EM::∼EM ( ) [inline]

```
{ } // end of default destruction.
```

5.1.2 Member Function Documentation

5.1.2.1 int gift::EM::EStep ( )

References EStepThread(), and gift::functionThread().

Referenced by trainEM().

```
                  {
    // both of them works.
    return gift::functionThread(&EM::EStepThread,thread,this);
    //return functionThread(&EM::EStepThread, thread);
  } // end of function
```

5.1.2.2 int gift::EM::EStep (int)

References iterdrugSub2ProteinSub(), and gift::observedDrug2ProteinMatrix.

```
                    {
    for(int i=0;i<drugNum;++i){
      for(int j=0;j<proteinNum;++j){
        double tmp = iterdrugSub2ProteinSub(i,j);
        observedDrug2ProteinMatrix[i][j] = (1-fn)*(1-tmp) + fp*tmp;
        //(*observedDrug2Protein).at(i).at(j) = (1-fn)*(1-tmp) + fp*tmp;
      } // end of for loop j
    } // end of for loop i
    return 0;
  } // end of function
```

5.1.2.3 void gift::EM::EStepThread (int *threadNth*)

References iterdrugSub2ProteinSub().

Referenced by EStep().

```
                                {
    //std::cout<<"This is thread "<<threadNth<<" for EStep..."<<std::endl;
    for(int i=threadNth;i<drugNum;i+=thread){
      for(int j=0;j<proteinNum;++j){
        double tmp = iterdrugSub2ProteinSub(i,j);
        (*observedDrug2Protein).at(i).at(j) = (1-fn)*(1-tmp) + fp*tmp;
      } // end of for loop j
    } // end of for loop i
    //return 0;
  } // end of function
```

5.1.2.4 int gift::EM::functionThread (void(EM::∗)(int) *function,* int *thread*) [inline]

```
                                                            {
      boost::thread *y;
      boost::thread_group * x = new boost::thread_group;
      for(int i=0;i<thread;++i){
        y = new boost::thread(function,this,i);
        x->add_thread(y);
      } // end of loop for i
      x->join_all();
      delete x;
      return 0;
    } // end of function.
```

5.1.2.5 double gift::EM::iterdrugSub2ProteinSub (int *drugIndex,* int *proteinIndex*)

Referenced by EStep(), EStepThread(), predictDrugs(), predictDrugsProteins(), predictProteins(), recLoglikely(), and varEM().

```
                                                                {
    double tmp = 0;
    if ((*drug2sub)[drugIndex].empty() || (*protein2sub)[proteinIndex].empty()){
      std::cerr<<"[ERROR]: some row of drug2sub or protein2sub is empty..."<<std::endl;
      return 1;
    } // end of if
    for(auto const & m : (*drug2sub)[drugIndex]){
      for(auto const & n : (*protein2sub)[proteinIndex]){
        tmp += log(1 - (*drugSub2proteinSub)[m][n]);
      } // end of loop n
    } // end of loop m
    return exp(tmp);
  } // end of function.
```

5.1.2.6 int gift::EM::MStep ( )

References gift::functionThread(), and MStepThread().

Referenced by trainEM().

```
                 {
    // both of them works.
    return gift::functionThread(&EM::MStepThread,thread,this);
    //return functionThread(&EM::MStepThread,thread);
  } // end of function
```

5.1.2.7 int gift::EM::MStep (int)

References gift::drugSub2proteinSubMatrix.

```
                {
  for(int i=0;i<subNum;++i){
    for(int j=0;j<domainNum;++j){
      double tmp = 0;
      for(auto  &m : (*sub2drug)[i]){
        for(auto  &n : (*sub2protein)[j]){
          double observed = (*observedDrug2Protein)[m][n];
          tmp = (std::find((*drug2protein)[m].begin(),(*drug2protein)[m].end(),n)
            != (*drug2protein)[m].end() ) ? (1-fn)/observed : fn/(1-observed);
        } // end of loop n
      } // end of loop m
      int tmpNum = (*sub2drug)[i].size() + (*sub2protein)[j].size();
      tmp = log((*drugSub2proteinSub)[i][j]) + log(tmp/tmpNum);
      drugSub2proteinSubMatrix[i][j] = exp(tmp);
      //        (*drugSub2proteinSub).at(i).at(j) = exp(tmp);
    } // end of loop j
  } // end of for loop i
  return 0;
} // end of function
```

5.1.2.8 void gift::EM::MStepThread (int *threadNth*)

Referenced by MStep().

```
                                 {
  //std::cout<<"This is thread "<<threadNth<<" for MStep..."<<std::endl;
  for(int i=threadNth;i<subNum;i+=thread){
    for(int j=0;j<domainNum;++j){
      double tmp = 0;
      for(auto const &m : (*sub2drug)[i]){
        for(auto const &n : (*sub2protein)[j]){
          double observed = (*observedDrug2Protein)[m][n];
          tmp = (std::find((*drug2protein)[m].begin(),(*drug2protein)[m].end(),n)
            != (*drug2protein)[m].end() ) ? (1-fn)/observed : fn/(1-observed);
        } // end of loop n
      } // end of loop m
      int tmpNum = (*sub2drug)[i].size() + (*sub2protein)[j].size();
      tmp = log((*drugSub2proteinSub)[i][j]) + log(tmp/tmpNum);
      (*drugSub2proteinSub)[i][j] = exp(tmp);
    } // end of loop j
  } // end of for loop i
} // end of function
```

5.1.2.9 int gift::EM::outTrainResult ( )

References gift::writeMatrix().

Referenced by main().

```
                        {
  writeMatrix(outDrugSub2ProteinSubFileName, *drugSub2proteinSub,
              outputDelims);
  return 0;
} // end of function
```

5.1.2.10 int gift::EM::outTrainVariance ( )

References gift::writeMatrix().

Referenced by main().

```
                           {
  writeMatrix(outVarDrugSub2proteinSubFileName, *vardrugSub2proteinSub,
              outputDelims);
  return 0;
} // end of function
```

5.1.2.11 int gift::EM::predictDrugs ( )

References gift::drugName2Index, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictDrugNameList, and gift::proteinNameList.

Referenced by predictEM().

```
                       {
  std::cout<<"Now predict given drugs against all the proteins, "
           <<"which are in our training data set." << std::endl;
  IntList predictDrugIndex;
  std::vector<double> tmpCalc;
  nameList existNameList;
  getIndexFromHash(drugName2Index,
predictDrugNameList,
                   predictDrugIndex, existNameList);
  if (existNameList.size() < 1) {
    std::cerr<<"No drug Index found, and quit." <<std::endl;
    return 1;
  } // end of if
  // output the result.
  std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
  if (!output.is_open()){
    std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
    return 1;
  } // end of if
  using boost::algorithm::join;
  using boost::adaptors::transformed;
  // print the first row as protein names.
  output<<"proteins"<<outputDelims;
  output<<join(proteinNameList,outputDelims)<<std::endl;
  int num = 0;
  for(const auto & drug : predictDrugIndex){
    for(int j=0;j<proteinNum;++j){
      tmpCalc.push_back(1 - iterdrugSub2ProteinSub(drug,j));
    } // end of loop for j
    output<<existNameList[num] << outputDelims;
    // transformed without static_cast should also work?
    output<<join(tmpCalc |
            transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                 outputDelims)
          << std::endl;
    tmpCalc.clear();
    ++num;
  } // end of loop for drug
  return 0;
} // end of functions.
```

5.1.2.12 int gift::EM::predictDrugsProteins ( )

References gift::drugName2Index, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictDrugNameList, gift::predictProteinNameList, and gift::proteinName2Index.

Referenced by predictEM().

```
                              {
  std::cout<<"Now predict given drugs against given proteins." << std::endl;
  IntList predictDrugIndex;
  IntList predictProteinIndex;
  std::vector<double> tmpCalc;
  nameList existdrugNameList;
  nameList existproteinNameList;
  getIndexFromHash(drugName2Index,
predictDrugNameList,
                   predictDrugIndex, existdrugNameList);
  getIndexFromHash(proteinName2Index,
predictProteinNameList,
                   predictProteinIndex, existproteinNameList);
  if (existdrugNameList.size()<1) {
    std::cerr<<"No drug Index found, and quit." <<std::endl;
    return 1;
  } // end of if
  if (existproteinNameList.size()<1) {
    std::cerr<<"No protein Index found, and quit." << std::endl;
    return 1;
  } // end of if
  // output the result.
  std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
  if (!output.is_open()){
    std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
    return 1;
  } // end of if
  using boost::algorithm::join;
  using boost::adaptors::transformed;
  // print the first row as protein names.
  output<<"Names"<<outputDelims;
  output<<join(existproteinNameList,outputDelims)<<std::endl;
  int num = 0;
  for(const auto & drug : predictDrugIndex){
    for(const auto & protein : predictProteinIndex){
      tmpCalc.push_back(1 - iterdrugSub2ProteinSub(drug,protein));
    } // end of loop for protein
    output<<existdrugNameList[num] <<outputDelims;
    // transformed without static_cast should also work?
    output<<join(tmpCalc |
            transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                 outputDelims)
          << std::endl;
    tmpCalc.clear();
    ++num;
  } // end of loop for drug
  return 0;
} // end of functions.
```

5.1.2.13 int gift::EM::predictDrugsProteinsWithSubs ( )

References gift::drugName2Index, gift::getIndexFromHash(), gift::predictDrugNameList, gift::predictProtein2Sub←List, and gift::predictProteinNameList_WithSubs.

Referenced by predictEM().

```
                                      {
  std::cout<<"Now predict given drugs against given proteins with subs."
           << std::endl;
  IntList predictDrugIndex;
  std::vector<double> tmpCalc;
  nameList existdrugNameList;
  getIndexFromHash(drugName2Index,
predictDrugNameList,
                   predictDrugIndex, existdrugNameList);
  if (existdrugNameList.size()<1) {
    std::cerr<<"No drug Index found, and quit." <<std::endl;
```

```
      return 1;
    } // end of if
    // output the result.
    std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
    if (!output.is_open()){
      std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
      return 1;
    } // end of if
    using boost::algorithm::join;
    using boost::adaptors::transformed;
    // print the first row as protein names.
    output<<"Names"<<outputDelims;
    output<<join(predictProteinNameList_WithSubs,outputDelims)<<std::endl;
    int num = 0;
    double tmp;
    for(const auto & drug : predictDrugIndex){
      for(int j=0;j<predictProteinNameList_WithSubs.size();++j){
        tmp = 0;
        for(auto const & m : (*drug2sub)[drug]){
          for(auto const & n : predictProtein2SubList[j]){
            tmp += log(1 - (*drugSub2proteinSub)[m][n]);
          } // end of loop n
        } // end of loop for m
        tmpCalc.push_back(1 - exp(tmp));
      } // end of loop for protein
      output<<existdrugNameList[num]<<outputDelims;
      // transformed without static_cast should also work?
      output<<join(tmpCalc |
              transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                   outputDelims)
            << std::endl;
      tmpCalc.clear();
      ++num;
    } // end of loop for drug
    return 0;
  } // end of functions.
```

5.1.2.14 int gift::EM::predictDrugsWithSubs ( )

References gift::predictDrug2SubList, gift::predictDrugNameList_WithSubs, and gift::proteinNameList.

Referenced by predictEM().

```
                              {
    std::cout<<"Now predict given drugs with subs against all the proteins, "
             <<"which are in our training data set." << std::endl;
    std::vector<double> tmpCalc;
    nameList existNameList;
    // output the result.
    std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
    if (!output.is_open()){
      std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
      return 1;
    } // end of if
    using boost::algorithm::join;
    using boost::adaptors::transformed;
    // print the first row as protein names.
    output<<"proteins"<<outputDelims;
    output<<join(proteinNameList,outputDelims)<<std::endl;
    double tmp;
    for(int i=0;i<predictDrugNameList_WithSubs.size();++i){
      for(int j=0;j<proteinNum;++j){
        tmp = 0;
        for(auto const & m : predictDrug2SubList[i]){
          for(auto const & n : (*protein2sub)[j]){
            tmp += log(1 - (*drugSub2proteinSub)[m][n]);
          } // end of loop for n
        } // end of loop for m
        tmpCalc.push_back(1 - exp(tmp));
      } // end of loop for j
      output<<predictDrugNameList_WithSubs[i]<<outputDelims;
      // transformed without static_cast should also work?
      output<<join(tmpCalc |
              transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                   outputDelims)
            << std::endl;
      tmpCalc.clear();
    } // end of loop for drug
    return 0;
  } // end of functions.
```

5.1.2.15 int gift::EM::predictDrugsWithSubsProteins ( )

References gift::getIndexFromHash(), gift::predictDrug2SubList, gift::predictDrugNameList_WithSubs, gift::predictProteinNameList, and gift::proteinName2Index.

Referenced by predictEM().

```
                                      {
  std::cout<<"Now predict given drugs with subs against given proteins."
           << std::endl;
  IntList predictProteinIndex;
  std::vector<double> tmpCalc;
  nameList existproteinNameList;
  getIndexFromHash(proteinName2Index,
predictProteinNameList,
                     predictProteinIndex, existproteinNameList);
  if (existproteinNameList.size()<1) {
    std::cerr<<"No protein Index found, and quit." <<std::endl;
    return 1;
  } // end of if
  // output the result.
  std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
  if (!output.is_open()){
    std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
    return 1;
  } // end of if
  using boost::algorithm::join;
  using boost::adaptors::transformed;
  // print the first row as protein names.
  output<<"Names"<<outputDelims;
  output<<join(existproteinNameList,outputDelims)<<std::endl;
  double tmp;
  for(int i=0;i<predictDrugNameList_WithSubs.size();++i){
    for(auto const protein : predictProteinIndex){
      tmp = 0;
      for(auto const & m : predictDrug2SubList[i]){
        for(auto const & n : (*protein2sub)[protein]){
          tmp += log(1 - (*drugSub2proteinSub)[m][n]);
        } // end of loop n
      } // end of loop for m
      tmpCalc.push_back(1 - exp(tmp));
    } // end of loop for protein
    output<<predictDrugNameList_WithSubs[i]<<outputDelims;
    // transformed without static_cast should also work?
    output<<join(tmpCalc |
            transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                 outputDelims)
          << std::endl;
    tmpCalc.clear();
  } // end of loop for i
  return 0;
} // end of functions.
```

5.1.2.16 int gift::EM::predictDrugsWithSubsProteinsWithSubs ( )

References gift::predictDrug2SubList, gift::predictDrugNameList_WithSubs, gift::predictProtein2SubList, and gift::predictProteinNameList_WithSubs.

Referenced by predictEM().

```
                                              {
  std::cout<<"Now predict given drugs with subs against given proteins with subs."
           << std::endl;
  std::vector<double> tmpCalc;
  // output the result.
  std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
  if (!output.is_open()){
    std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
    return 1;
  } // end of if
  using boost::algorithm::join;
  using boost::adaptors::transformed;
  // print the first row as protein names.
  output<<"Names"<<outputDelims;
```

```
    output<<join(predictProteinNameList_WithSubs,outputDelims)<<std::endl;
    double tmp;
    for(int i=0;i<predictDrugNameList_WithSubs.size();++i){
      for(int j=0;j<predictProteinNameList_WithSubs.size();++j){
        tmp = 0;
        for(auto const & m : predictDrug2SubList[i]){
          for(auto const & n : predictProtein2SubList[j]){
            tmp += log(1 - (*drugSub2proteinSub)[m][n]);
          } // end of loop n
        } // end of loop for m
        tmpCalc.push_back(1 - exp(tmp));
      } // end of loop for protein
      output<<predictDrugNameList_WithSubs[i]<<outputDelims;
      // transformed without static_cast should also work?
      output<<join(tmpCalc |
              transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                   outputDelims)
            << std::endl;
      tmpCalc.clear();
    } // end of loop for i
    return 0;
  } // end of functions.
```

5.1.2.17 int gift::EM::predictEM ( )

References gift::predictDrugNameList, gift::predictDrugNameList_WithSubs, predictDrugs(), predictDrugs↩ Proteins(), predictDrugsProteinsWithSubs(), predictDrugsWithSubs(), predictDrugsWithSubsProteins(), predict↩ DrugsWithSubsProteinsWithSubs(), gift::predictProteinNameList, gift::predictProteinNameList_WithSubs, predict↩ Proteins(), and predictProteinsWithSubs().

Referenced by main().

```
                   {
    // Note: we only run one situation one time:
    // - Provide both drugs and proteins Names.
    // - Provide both drugs and proteins with Subs.
    // - Provide both drugs Names and proteins with Subs.
    // - Provide both drugs with Subs and proteins Names.
    // - Provide only drugs Names.
    // - Provide only drug with Subs.
    // - Provide only protein Names.
    // - Provide only protein with Subs.
    std::cout<<"Now Run predictEM for task: predict..." << std::endl;

    if (!predictDrugNameList_WithSubs.empty() &
        !predictProteinNameList_WithSubs.empty()){
      predictDrugsWithSubsProteinsWithSubs();
    }  else if (!predictDrugNameList_WithSubs.empty() &
        !predictProteinNameList.empty()){
      predictDrugsWithSubsProteins();
    } else if (!predictProteinNameList_WithSubs.empty() &
        !predictDrugNameList.empty()){
      predictDrugsProteinsWithSubs();
    } else if (!predictDrugNameList.empty() &
        !predictProteinNameList.empty()) {
      predictDrugsProteins();
    } else if (!predictDrugNameList.empty()){
      predictDrugs();
    } else if (!predictDrugNameList_WithSubs.empty()){
      predictDrugsWithSubs();
    } else if (!predictProteinNameList.empty()){
      predictProteins();
    } else if (!predictProteinNameList_WithSubs.empty()){
      predictProteinsWithSubs();
    } else {
      std::cerr<<"No files for task: predict, and quit."<< std::endl;
      return 1;
    } // end of if else if else.
    return 0;
  } // end of function
```

5.1.2.18 int gift::EM::predictProteins ( )

References gift::drugNameList, gift::getIndexFromHash(), iterdrugSub2ProteinSub(), gift::predictProteinNameList, and gift::proteinName2Index.

Referenced by predictEM().

```
                         {
    std::cout<<"Now predict given proteins against all the drugs, "
             <<"which are in our training data set." << std::endl;
    IntList predictProteinsIndex;
    std::vector<double> tmpCalc;
    nameList existNameList;
    getIndexFromHash(proteinName2Index,
  predictProteinNameList,
                     predictProteinsIndex, existNameList);
    if (existNameList.size() < 1) {
      std::cerr<<"No protein Index found, and quit." <<std::endl;
      return 1;
    } // end of if
    // output the result.
    std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
    if (!output.is_open()){
      std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
      return 1;
    } // end of if
    using boost::algorithm::join;
    using boost::adaptors::transformed;
    // print the first row as drug names.
    output<<"drugs"<<outputDelims;
    output<<join(drugNameList,outputDelims)<<std::endl;
    int num = 0;
    for(const auto & protein : predictProteinsIndex){
      for(int j=0;j<drugNum;++j){
        tmpCalc.push_back(1 - iterdrugSub2ProteinSub(j,protein));
      } // end of loop for j
      output<<existNameList[num]<<outputDelims;
      // transformed without static_cast should also work?
      output<<join(tmpCalc |
              transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                   outputDelims)
           << std::endl;
      tmpCalc.clear();
      ++num;
    } // end of loop for protein
    return 0;
  } // end of functions.
```

5.1.2.19 int gift::EM::predictProteinsWithSubs ( )

References gift::drugNameList, gift::predictProtein2SubList, and gift::predictProteinNameList_WithSubs.

Referenced by predictEM().

```
                                 {
    std::cout<<"Now predict given proteins with subs against all the drugs, "
             <<"which are in our training data set." << std::endl;
    IntList predictProteinsIndex;
    std::vector<double> tmpCalc;
    nameList existNameList;
    // output the result.
    std::ofstream output (outPredictCPIsFileName,std::ofstream::out);
    if (!output.is_open()){
      std::cerr<<"Error open file "<<outPredictCPIsFileName<<std::endl;
      return 1;
    } // end of if
    using boost::algorithm::join;
    using boost::adaptors::transformed;
    // print the first row as drug names.
    output<<"drugs"<<outputDelims;
    output<<join(drugNameList,outputDelims)<<std::endl;
    double tmp;
    for(int i=0;i<predictProteinNameList_WithSubs.size();++i){
      for(int j=0;j<drugNum;++j){
```

```
      tmp = 0;
      for(auto const & m : (*drug2sub)[j]){
        for(auto const & n : predictProtein2SubList[i]){
          tmp += log(1 - (*drugSub2proteinSub)[m][n]);
        } // end of loop for n
      } // end of loop for m
      tmpCalc.push_back(1 - exp(tmp));
    } // end of loop for j
    output<<predictProteinNameList_WithSubs[i]<<outputDelims;
    // transformed without static_cast should also work?
    output<<join(tmpCalc |
            transformed(static_cast<std::string(*)(double)>(std::to_string) ),
                 outputDelims)
          << std::endl;
    tmpCalc.clear();
  } // end of loop for protein
  return 0;
} // end of functions.
```

5.1.2.20 double gift::EM::recLoglikely ( )

References iterdrugSub2ProteinSub().

Referenced by trainEM().

```
                         {
  double loglikely = 0;
  double tmp;
  std::vector<int>::iterator it;
  for(int i=0;i<drugNum;++i) {
    for(int j=0;j<proteinNum;++j) {
      tmp = iterdrugSub2ProteinSub(i,j);
      it = std::find((*drug2protein)[i].begin(),
                     (*drug2protein)[i].end(),j);
      loglikely += it==(*drug2protein)[i].end() ?
        log(1-(1-fn)*(1-tmp)-fp*tmp) : log((1-fn)*(1-tmp) + fp*tmp);
    } // end of loop j
  } // end of loop i
  return loglikely;
} // end of function.
```

5.1.2.21 int gift::EM::setLoglikely (double *logscore*) [inline]

Referenced by trainEM().

```
                                           {
    (*loglikely).push_back(logscore);
    return 0;
  } // end of function
```

5.1.2.22 int gift::EM::setPointerDrug2Protein (IntArrayList & *d2p*) [inline]

```
                                                         {
    drug2protein = &d2p;
    return 0;
  } // end of func
```

5.1.2.23 int gift::EM::setPointerDrug2Sub (IntArrayList & *d2s*) [inline]

```
                                                     {
    drug2sub = &d2s;
    return 0;
  } // end of func
```

5.1.2.24 int gift::EM::setPointerDrugSub2ProteinSub (numericMatrix & *ds2ps*) [inline]

```
                                                                    {
      drugSub2proteinSub = &ds2ps;
      return 0;
    } // end of func
```

5.1.2.25 int gift::EM::setPointerProtein2Sub (IntArrayList & *p2s*) [inline]

```
                                                          {
      protein2sub = &p2s;
      return 0;
    } // end of func
```

5.1.2.26 int gift::EM::trainEM ( )

References EStep(), MStep(), gift::recLogLeastNum, recLoglikely(), and setLoglikely().

Referenced by main().

```
                  {
    // lack of loglikely record
    for(int i=0;i<iterNum;++i){
      std::cout<<"Current iteration number is " << i << std::endl;
      std::chrono::steady_clock::time_point tBegin =
        std::chrono::steady_clock::now();
      EStep();
      //std::cout<<"EStep Testing..."<<std::endl;
      //EStep(1); // for testEM
      std::chrono::steady_clock::time_point tEnd =
        std::chrono::steady_clock::now();
      std::cout<<"EStep Time difference (s): "
       <<std::chrono::duration_cast<std::chrono::seconds>(tBegin-tEnd).count()
               <<std::endl;

      tBegin = std::chrono::steady_clock::now();
      MStep();
      //MStep(1); // for testEM
      tEnd = std::chrono::steady_clock::now();
      std::cout<<"MStep Time difference (s): "
       <<std::chrono::duration_cast<std::chrono::seconds>(tBegin-tEnd).count()
               <<std::endl;

      if (loglikelyRecord || i<=recLogLeastNum || i >= iterNum-recLogLeastNum) {
        double tmplog = recLoglikely();
        std::cout<<"Current loglikelyhood is " << tmplog << std::endl;
        // // for test
        // std::cout<<"Current observedDrug2Protein Matrix is: "<<std::endl;
        // printMatrix(*observedDrug2Protein);
        // std::cout<<"Current drugSub2proteinSub Matrix is: "<<std::endl;
        // printMatrix(*drugSub2proteinSub);

        setLoglikely(tmplog);
      } // end of if
    } // end of loop i
    std::chrono::system_clock::time_point endTime =
      std::chrono::system_clock::now();
    std::time_t endTimeT = std::chrono::system_clock::to_time_t(endTime);
    std::cout<<"Finished computation at " << std::ctime(&endTimeT) << std::endl;
    return 0;
  } // end of function
```

5.1.2.27 int gift::EM::varEM ( )

References iterdrugSub2ProteinSub().

Referenced by main().

```
                {
    numericMatrix quesiDev;
    std::vector<double> tmpDev;
    // Note: not check drugSub2proteinsub values larger than 0.95.
    for (int i=0;i<drugNum;++i) {
      for (int j=0;j<proteinNum;++j){
        tmpDev.push_back(iterdrugSub2ProteinSub(i,j));
      } // end of loop j
      quesiDev.push_back(tmpDev);
      tmpDev.clear();
    } // end of loop i

    for(int i=0;i<subNum;++i){
      for(int j=0;j<domainNum;++j){
        double tmp_t = 0;
        //std::vector<double> tmp_t_array;
        double tmp_s = 0;
        //std::vector<double> tmp_s_array;
        double tmpLikely = (*observedDrug2Protein)[i][j];
        double tmp_sum = 0;
        for(auto const & m : (*sub2drug)[i]){
          for(auto const & n : (*sub2protein)[j]){
            tmp_t = std::find((*drug2protein)[m].begin(),(*drug2protein)[m].end(),n)
              == (*drug2protein)[m].end() ? 1/pow(1-tmpLikely,2.0) : 1/pow(tmpLikely,2.0);
            // tmp_t_array.push_back(tmp_t);
            tmp_s = (1-fn-fp) * quesiDev[m][n] / (1 - (*drugSub2proteinSub)[i][j]);
            //tmp_s_array.push_back(tmp_s);
            tmp_sum += pow(tmp_s,2.0)*tmp_t;
          } // end of loop n
        } // end of loop m
        (*vardrugSub2proteinSub)[i][j] = tmp_sum;
      } // end of loop j
    } // end of loop i
    return 0;
  } // end of function
```

The documentation for this class was generated from the following files:

- gift.hpp
- EMGift.cpp

5.2 gift::parameters Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- parameters (const std::string) throw (std::string)
- int setDrugNum (int number)
- int setSubNum (int number)
- int setProteinNum (int number)
- int setDomainNum (int number)
- int InitDrugSub2ProteinSub ()
- int InitVarDrugSub2ProteinSub ()
- int InitDrugName2Index ()
- int InitProteinName2Index ()
- int InitDrugSubNameList ()
- int InitProteinSubNameList ()
- int InitPredictParameters () throw (std::string)
- int InitObservedDrug2ProteinMatrix ()

Public Attributes

- std::string drug2proteinFileName
- std::string drug2subFileName
- std::string protein2subFileName
- std::string drugSub2proteinSubFileName
- std::string drugNameListFile
- std::string drugSubNameListFile
- std::string proteinNameListFile
- std::string proteinSubNameListFile
- bool loglikelyRecord
- double alphaEB
- double betaEB
- double fn
- double fp
- int thread
- int iterNum
- int drugNum
- int subNum
- int domainNum
- int proteinNum
- std::string inputDelims
- std::string task
- std::string chemfpRec
- std::string proteinfpRec
- std::string CPIsRec
- std::string predictDrugsFileName
- std::string predictProteinsFileName
- std::string predictDrugsFileName_WithSubs
- std::string predictProteinsFileName_WithSubs
- std::string outputDelims
- std::string outRecordFileName
- std::string outPredictCPIsFileName
- std::string outDrugSub2ProteinSubFileName
- std::string outVarDrugSub2proteinSubFileName

5.2.1 Constructor & Destructor Documentation

5.2.1.1 gift::parameters::parameters (const std::string *configFile*) throw std::string)

References gift::BoolToString(), gift::rowCol::colNum, gift::Matrix2Fingerpints(), gift::Matrix2FingerprintsBy↩
Column(), gift::printIntArrayList(), gift::rowColFile(), and gift::rowCol::rowNum.

```
                                                                {
    std::cout<<"Now set parameters with configFile."<<std::endl;
    // use boost program_options to read configs from a given file.
    namespace po = boost::program_options;
    po::options_description desc("GIFT Parameter options");
    desc.add_options()
      // input data file name
      ("drug2proteinFileName",
       po::value<std::string>(&drug2proteinFileName),
       "file name for drug protein interactions")
      ("drug2subFileName",po::value<std::string>(&drug2subFileName),
       "file name for drug to substructure")
      ("protein2subFileName",po::value<std::string>(&protein2subFileName),
       "file name for protein to substructure")
      ("drugSub2proteinSubFileName",
       po::value<std::string>(&drugSub2proteinSubFileName),
       "file name for drugSub to proteinSub interaction probability")
      // input name list file name
      ("drugNameListFile",po::value<std::string>(&drugNameListFile),
       "file name for drug names")
      ("drugSubNameListFile",po::value<std::string>(&drugSubNameListFile),
       "file name for drug substructures names")
      ("proteinNameListFile",po::value<std::string>(&proteinNameListFile),
       ("file name for protein names"))
      ("proteinSubNameListFile",po::value<std::string>(&proteinSubNameListFile),
       "file name for protein substructures names")
      // input parameters for EM
      ("alphaEB",po::value<double>(&alphaEB)->default_value(0.05),
       "parameter for Empirical Bayesian estimates for initEM")
      ("betaEB",po::value<double>(&betaEB)->default_value(0.05),
       "parameter for Empirical Bayesian estimates for initEM")
      ("fp", po::value<double>(&fp)->default_value(0.85),
       "false positive rate")
      ("fn", po::value<double>(&fn)->default_value(0.0001),
       "false negative rate")
      ("threadNum", po::value<int>(&thread)->default_value(1),
       "thread number for EM")
      ("EMIterationNum", po::value<int>(&iterNum)->default_value(300),
       "iteration numbers for EM")
      ("task", po::value<std::string>(&task)->default_value("train"),
       "run gift for train or predict")
      ("loglikelyRecord",
       po::value<bool>(&loglikelyRecord)->default_value(false),
       "whether or not to record the loglikely in every step")
      ("inputDelims",po::value<std::string>(&inputDelims)->default_value("\t,"),
       "sep character for input files")
      // input file version information.
      ("chemFingerPrintRecord",
       po::value<std::string>(&chemfpRec)->default_value("ComFP: PUBCHEM"),
       "source and version of chemical fingerprints")
      ("proteinFingerPrintRecord",
       po::value<std::string>(&proteinfpRec)->default_value("Pfam: 2011-07"),
       "source and version of protein fingerprints/domains")
      ("comProteinInteractionRecord",
       po::value<std::string>(&CPIsRec)->default_value("DrugBank: 2011-07"),
       "source and version of compound-protien interactions")
      // input file names for prediction
      ("predictDrugsFileName",po::value<std::string>(&predictDrugsFileName),
       "file name for drug names used for prediciton by gift")
      ("predictProteinFileName",po::value<std::string>(&predictProteinsFileName),
       "file name for protein names used for prediction by gift")
       ("predictDrugsFileName_WithSubs",
        po::value<std::string>(&predictDrugsFileName_WithSubs),
        "file name for drugs names together with their substructures.")
       ("predictProteinsFileName_WithSubs",
        po::value<std::string>(&predictProteinsFileName_WithSubs),
        "file name for protein names together with their substructures.")
       // output file name and format
       ("outputDelims",po::value<std::string>(&outputDelims)->default_value("\t,"),
        "sep character for output files")
       ("outRecordFileName",
        po::value<std::string>(&outRecordFileName)->default_value("CPIs"),
        "file name for output records")
       ("outPredictCPIsFileName",
        po::value<std::string>(&outPredictCPIsFileName),
        "file name for output CPIs")
       ("outDrugSub2ProteinSubFileName",
        po::value<std::string>(&outDrugSub2ProteinSubFileName),
        "file name for output drugSub2proteinSub")
       ("outVarDrugSub2proteinSubFileName",
        po::value<std::string>(&outVarDrugSub2proteinSubFileName),
        "file name for output variance of drugSub2proteinSub");
     po::variables_map vm;
     std::ifstream input (configFile, std::ios::in);
     po::store(po::parse_config_file(input, desc), vm);
     po::notify(vm);
     // input.exceptions(std::ifstream::failbit | std::ifstream::badbit);
     // try {
     //   input.open(configFile, std::ifstream::in);
     //   if (input.peek() == std::ifstream::traits_type::eof()) {
     //     std::cerr<<configFile<<" is empty." << std::endl;
     //   } // end of if.
     //   po::store(po::parse_config_file(input, desc), vm);
     //   po::notify(vm);
     // } catch (std::ifstream::failure e) {
     //   std::cerr <<"Exceptions open/read file "<<configFile<<std::endl;
     // } // end of try catch
     // default training data parameters.
     // they will be set when read data files.

     rowCol tmp;
     rowColFile(drug2subFileName,tmp,inputDelims);
```

```
  drugNum = tmp.rowNum;
  subNum = tmp.colNum;
  std::cout<<"Drug Number is "<<drugNum<<std::endl;
  std::cout<<"DrugSub Number is "<<subNum<<std::endl;
  rowColFile(protein2subFileName,tmp,inputDelims);
  domainNum = tmp.colNum;
  proteinNum = tmp.rowNum;
  std::cout<<"Protein Number is "<<proteinNum<<std::endl;
  std::cout<<"ProteinSub Number is "<<domainNum<<std::endl;


  // load global data for gift.
  Matrix2Fingerpints(drug2proteinFileName,
drug2proteinList,inputDelims);
  //printIntArrayList(drug2proteinList); // for test
  Matrix2Fingerpints(protein2subFileName,
protein2domainList,inputDelims);
  //printIntArrayList(protein2domainList); // for test
  Matrix2Fingerpints(drug2subFileName,
drug2subList,inputDelims);
  //printIntArrayList(drug2subList); // for test

  IntList tmpIntArray;
  std::cout<<"Initialize the sub2drugList..."<<std::endl;
  Matrix2FingerprintsByColumn(drug2subFileName,
sub2drugList,subNum,inputDelims);
  std::cout<<"Finish the init of sub2drugList."<<std::endl;
  printIntArrayList(sub2drugList); // for test

  std::cout<<"Initialize the domain2proteinList..."<<std::endl;
  Matrix2FingerprintsByColumn(protein2subFileName,
domain2proteinList,
                              domainNum, inputDelims);
  std::cout<<"Finish the init of domain2proteinList."<<std::endl;
  printIntArrayList(domain2proteinList); // for test

  InitDrugSub2ProteinSub();
  InitVarDrugSub2ProteinSub();

  InitObservedDrug2ProteinMatrix();
  // load NameList.
  InitDrugName2Index();
  InitProteinName2Index();
  InitDrugSubNameList();
  InitProteinSubNameList();

  // load predicted name list and possible subs if task is prediction.
  InitPredictParameters(); // throw std::string.

  // print the setting results.
  std::cout<<"parameters have been set."<<std::endl;
  for (const auto& it : vm){
    std::cout<< it.first.c_str() << ": ";
    auto& value = it.second.value(); // return boost::any reference type.
    // any_cast use the any * as input and return the pointer with type infor.
    if (auto v = boost::any_cast<int>(&value) ) {
      std::cout<< *v <<std::endl;
    } else if (auto v = boost::any_cast<double>(&value) ) {
      std::cout<< *v <<std::endl;
    } else if (auto v = boost::any_cast<bool>(&value) ) {
      std::cout<< BoolToString(*v) <<std::endl;
    } else if (auto v = boost::any_cast<std::string>(&value) ) {
      std::cout<< *v <<std::endl;
    } else {
      std::cout<< "Error type"<<std::endl;
    } // end of if
  } // end of for
  std::cout<<"drugNum: " <<drugNum<<std::endl;
  std::cout<<"subNum: "<<subNum<<std::endl;
  std::cout<<"domainNum: "<<domainNum<<std::endl;
  std::cout<<"proteinNum: "<<proteinNum<<std::endl;
} // end of class parameter constructor.
```

5.2.2 Member Function Documentation

5.2.2.1 int gift::parameters::InitDrugName2Index ( )

References drugNameListFile, gift::readName2IndexHash(), and gift::readNameListFromFile().

```
                                  {
    std::cout<<"Initialize DrugName2Index Hash..."<<std::endl;
    readNameListFromFile(drugNameListFile,
  drugNameList);
    readName2IndexHash(drugNameList,drugName2Index);
    std::cout<<"Finish DrugName2Index Hash."<<std::endl;
    return 0;
  } // end of function
```

5.2.2.2 int gift::parameters::InitDrugSub2ProteinSub ( )

References alphaEB, betaEB, domainNum, drugSub2proteinSubFileName, inputDelims, gift::readMatrix(), sub↩ Num, and task.

```
                                     {
    // This function must be run after class parameter initionlization.
    std::cout<< "Initialize the drugSub2proteinSub Matrix." << std::endl;
    if (task.compare("predict") == 0 ) {
      readMatrix(drugSub2proteinSubFileName,
  drugSub2proteinSubMatrix,
                 inputDelims);
      std::cout<< "Finish: read from file."<<std::endl;
    } else {
      std::vector<double> assoTmp; // temp array based on the assocaiton method.
      int N;
      int subNumTmp;
      int I = 0;
      std::vector<int>::iterator it;
      for (int i=0;i<subNum;++i){
        subNumTmp = sub2drugList[i].size();
        for (int j=0;j<domainNum;++j){
          N = domain2proteinList[j].size() * subNumTmp;
          for (const auto drug : sub2drugList[i]){
            for (const auto protein : domain2proteinList[j]){
              it = std::find(drug2proteinList[drug].begin(),
                             drug2proteinList[drug].end(), protein);
              I += (it==drug2proteinList[drug].end() ? 0 : 1);
            } // end of loop protein
          } // end of loop drug
          // revise association method with Emiprical Bayes.
          assoTmp.push_back((I+alphaEB)/(alphaEB+betaEB+N));
          I = 0;
        } // end of loop j
        drugSub2proteinSubMatrix.push_back(assoTmp);
        assoTmp.clear();
      } // end of loop i
      std::cout<<"Finish: initialize with associatiom method and emprical Bayes."
               <<std::endl;
    } // end of if else
    return 0;
  } // end of function
```

5.2.2.3 int gift::parameters::InitDrugSubNameList ( )

References drugSubNameListFile, and gift::readNameListFromFile().

```
                                   {
    readNameListFromFile(drugSubNameListFile,
  drugSubNameList);
    return 0;
  } // end of function
```

5.2.2.4 int gift::parameters::InitObservedDrug2ProteinMatrix ( )

References drugNum, proteinNum, and task.

```
                                               {
    std::cout<<"Init ObservedDrug2ProteinMatrix ..."<<std::endl;
    if(task.compare("predict") == 0){
      std::cout<<"Job is to predict, and ignore the inition of observedDrug2Protein."
               <<std::endl;
    } else{
      std::vector<double> tmpArray(proteinNum,0.1);
      for(int i=0;i<drugNum;++i){
        observedDrug2ProteinMatrix.push_back(tmpArray);
      } // end of loop for i.
    } // end of if else
    std::cout<<"Finish the init of observedDrug2ProteinMatrix."<<std::endl;
    return 0;
  } // end of function.
```

5.2.2.5 int gift::parameters::InitPredictParameters ( ) throw std::string)

References predictDrugsFileName, predictDrugsFileName_WithSubs, predictProteinsFileName, predictProteins←FileName_WithSubs, gift::readNameListFromFile(), and gift::readNameMatrixFromFile().

```
                                                        {
    // when task is predict, we use this function to init the corresponding
    // parameters.
    bool checkStatus = false;
    if (!predictDrugsFileName.empty()){
      checkStatus = true;
      readNameListFromFile(predictDrugsFileName,
  predictDrugNameList);
    } // end of if for drugfile
    if (!predictProteinsFileName.empty()){
      checkStatus = true;
      readNameListFromFile(predictProteinsFileName,
  predictProteinNameList);
    } // end of if for proteinfile
    if (!predictDrugsFileName_WithSubs.empty()){
      checkStatus  = true;
      // NEED FUNCTION.
      readNameMatrixFromFile(predictDrugsFileName_WithSubs
  ,
                              predictDrugNameList_WithSubs,
                              predictDrug2SubList);
    } // end of if for drug_withsubs file.
    if (!predictProteinsFileName_WithSubs.empty()){
      checkStatus = true;
      // NEED FUNCTION.
      readNameMatrixFromFile(
  predictProteinsFileName_WithSubs,
                              predictProteinNameList_WithSubs,
                              predictProtein2SubList);
    } // end of if for protein_withsubs file.
    if (!checkStatus){
      throw("Task for prediction, but no predicted files!");
    } // end of if for checkStatus.
    return 0;
  } // end of function
```

5.2.2.6 int gift::parameters::InitProteinName2Index ( )

References proteinNameListFile, gift::readName2IndexHash(), and gift::readNameListFromFile().

```
                                    {
    readNameListFromFile(proteinNameListFile,
  proteinNameList);
    readName2IndexHash(proteinNameList,
  proteinName2Index);
    return 0;
  } // end of function
```

5.2.2.7 int gift::parameters::InitProteinSubNameList ( )

References proteinSubNameListFile, and gift::readNameListFromFile().

```
                                         {
    readNameListFromFile(proteinSubNameListFile,
  proteinSubNameList);
    return 0;
  } // end of function
```

5.2.2.8 int gift::parameters::InitVarDrugSub2ProteinSub ( )

References domainNum, subNum, and task.

```
                                           {
    if (task.compare("predict") == 0 ) {
      std::cout<< "Job is to predict, skip init variance matrix." << std::endl;
    } else {
      std::cout<<"Initialize the variance matrix for drugSub2ProteinSub..."
               <<std::endl;
      std::vector<double> tmpArray (domainNum,1.01);
      for(int i=0;i<subNum;++i){
        vardrugSub2proteinSubMatrix.push_back(tmpArray);
      } // end of loop for i.
    } // end of if else
    std::cout<<"Finish init of variance matrix for drugSub2ProteinSub." <<std::endl;
    return 0;
  } // end of function
```

5.2.2.9 int gift::parameters::setDomainNum (int *number*) [inline]

```
{ domainNum = number; return 0; }
```

5.2.2.10 int gift::parameters::setDrugNum (int *number*) [inline]

```
{ drugNum = number; return 0; }
```

5.2.2.11 int gift::parameters::setProteinNum (int *number*) [inline]

```
{proteinNum = number; return 0; }
```

5.2.2.12 int gift::parameters::setSubNum (int *number*) [inline]

```
{ subNum = number; return 0; }
```

5.2.3 Member Data Documentation

5.2.3.1 double gift::parameters::alphaEB

Referenced by InitDrugSub2ProteinSub().

5.2.3.2 double gift::parameters::betaEB

Referenced by InitDrugSub2ProteinSub().

5.2.3.3 std::string gift::parameters::chemfpRec

5.2.3.4 std::string gift::parameters::CPIsRec

5.2.3.5 int gift::parameters::domainNum

Referenced by InitDrugSub2ProteinSub(), and InitVarDrugSub2ProteinSub().

5.2.3.6 std::string gift::parameters::drug2proteinFileName

5.2.3.7 std::string gift::parameters::drug2subFileName

5.2.3.8 std::string gift::parameters::drugNameListFile

Referenced by InitDrugName2Index().

5.2.3.9 int gift::parameters::drugNum

Referenced by InitObservedDrug2ProteinMatrix().

5.2.3.10 std::string gift::parameters::drugSub2proteinSubFileName

Referenced by InitDrugSub2ProteinSub().

5.2.3.11 std::string gift::parameters::drugSubNameListFile

Referenced by InitDrugSubNameList().

5.2.3.12 double gift::parameters::fn

5.2.3.13 double gift::parameters::fp

5.2.3.14 std::string gift::parameters::inputDelims

Referenced by InitDrugSub2ProteinSub().

5.2.3.15 int gift::parameters::iterNum

5.2.3.16 bool gift::parameters::loglikelyRecord

Referenced by gift::outRecord().

5.2.3.17 std::string gift::parameters::outDrugSub2ProteinSubFileName

5.2.3.18 std::string gift::parameters::outPredictCPIsFileName

5.2.3.19 std::string gift::parameters::outputDelims

5.2.3.20 std::string gift::parameters::outRecordFileName

Referenced by gift::outRecord().

5.2.3.21 std::string gift::parameters::outVarDrugSub2proteinSubFileName

5.2.3.22 std::string gift::parameters::predictDrugsFileName

Referenced by InitPredictParameters().

5.2.3.23 std::string gift::parameters::predictDrugsFileName_WithSubs

Referenced by InitPredictParameters().

5.2.3.24 std::string gift::parameters::predictProteinsFileName

Referenced by InitPredictParameters().

5.2.3.25 std::string gift::parameters::predictProteinsFileName_WithSubs

Referenced by InitPredictParameters().

5.2.3.26 std::string gift::parameters::protein2subFileName

5.2.3.27 std::string gift::parameters::proteinfpRec

5.2.3.28 std::string gift::parameters::proteinNameListFile

Referenced by InitProteinName2Index().

5.2.3.29 int gift::parameters::proteinNum

Referenced by InitObservedDrug2ProteinMatrix().

5.2.3.30 std::string gift::parameters::proteinSubNameListFile

Referenced by InitProteinSubNameList().

5.2.3.31 int gift::parameters::subNum

Referenced by InitDrugSub2ProteinSub(), and InitVarDrugSub2ProteinSub().

5.2.3.32 std::string gift::parameters::task

Referenced by InitDrugSub2ProteinSub(), InitObservedDrug2ProteinMatrix(), InitVarDrugSub2ProteinSub(), main(), and gift::outRecord().

5.2.3.33 int gift::parameters::thread

The documentation for this class was generated from the following files:

- gift.hpp
- parameterGift.cpp

5.3 gift::rowCol Class Reference

```
#include <gift.hpp>
```

Public Member Functions

- rowCol (int row, int col)
- rowCol ()

Public Attributes

- int rowNum
- int colNum

5.3.1 Constructor & Destructor Documentation

5.3.1.1 gift::rowCol::rowCol (int *row*, int *col*) [inline]

```
118 : rowNum(row), colNum(col) {}
```

5.3.1.2 gift::rowCol::rowCol ( ) [inline]

```
: rowNum(1), colNum(1) {}
```

5.3.2 Member Data Documentation

5.3.2.1 int gift::rowCol::colNum

Referenced by gift::parameters::parameters(), and gift::rowColFile().

5.3.2.2 int gift::rowCol::rowNum

Referenced by gift::parameters::parameters(), and gift::rowColFile().

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

- gift.hpp

Chapter 6

File Documentation

6.1 EMGift.cpp File Reference

```
#include <cmath>
#include <ctime>
#include <chrono>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/join.hpp>
#include <boost/range/adaptor/transformed.hpp>
#include "gift.hpp"
```

Namespaces

- gift

6.2 functionsGift.cpp File Reference

```
#include <ctime>
#include <chrono>
#include <boost/algorithm/string.hpp>
#include <boost/algorithm/string/join.hpp>
#include <boost/range/adaptor/transformed.hpp>
#include "gift.hpp"
```

Namespaces

- gift

Functions

- int gift::Matrix2Fingerpints (const std::string inputFile, IntArrayList &getFp, std::string delims)
- int gift::readMatrix (const std::string inputFile, numericMatrix &getMat, std::string delims)
- int gift::rowColFile (const std::string inputFile, rowCol &matrixRec, std::string delims)
- int gift::writeMatrix (const std::string outFileName, numericMatrix &resultMat, std::string delims)
- int gift::printIntArrayList (const IntArrayList &fromIntArrayList)
- int gift::printMatrix (const numericMatrix &fromMatrix)
- int gift::readNameListFromFile (const std::string inputFile, nameList &tonameList)
- int gift::readNameMatrixFromFile (const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std::string delims)
- int gift::readName2IndexHash (const nameList fromNameList, name2IndexHash &name2Index)
- int gift::getIndexFromHash (const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIndexList, nameList &existNameList)
- int gift::helpGift ()
- int gift::outRecord (parameters &EMparameters)
- int gift::Matrix2FingerprintsByColumn (const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)

6.3 gift.hpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <map>
#include <boost/thread/thread.hpp>
#include <boost/bind.hpp>
```

Classes

- class gift::rowCol
- class gift::parameters
- class gift::EM

Namespaces

- gift

Typedefs

- typedef std::vector< int > gift::IntList
- typedef std::vector< std::vector< int > > gift::IntArrayList
- typedef std::vector< std::vector< double > > gift::numericMatrix
- typedef std::map< std::string, int > gift::name2IndexHash
- typedef std::vector< std::string > gift::nameList

Functions

- const std::string gift::author ("Songpeng Zu")
- const std::string gift::email ("zusongpeng@gmail.com")
- const std::string gift::version ("gift-2.0")
- const std::string gift::updateTime ("2016-03-06")
- int gift::Matrix2Fingerpints (const std::string inputFile, IntArrayList &getFp, std::string delims)
- int gift::Matrix2FingerprintsByColumn (const std::string inputFile, IntArrayList &getFP, int rowNum, std::string delims)
- int gift::writeMatrix (const std::string outFileName, numericMatrix &resultMat, std::string delims)
- int gift::printIntArrayList (const IntArrayList &fromIntArrayList)
- int gift::printMatrix (const numericMatrix &fromMatrix)
- int gift::readMatrix (const std::string inputFile, numericMatrix &getMat, std::string delims)
- int gift::readNameListFromFile (const std::string inputFile, nameList &tonameList)
- int gift::readNameMatrixFromFile (const std::string inputFile, nameList &tonameList, IntArrayList &getFP, std::string delims)
- int gift::readName2IndexHash (const nameList fromNameList, name2IndexHash &name2Index)
- int gift::getIndexFromHash (const name2IndexHash &name2Index, const nameList fromNameList, IntList &toIndexList, nameList &existNameList)
- int gift::rowColFile (const std::string inputFile, rowCol &matrixRec, std::string delims)
- int gift::helpGift ()
- int gift::outRecord (parameters &EMparameters)
- template< typename func >
  int gift::functionThread (func useFun, int thread, EM ∗point)

Variables

- const int gift::recLogLeastNum = 5
- IntArrayList gift::drug2proteinList
- IntArrayList gift::drug2subList
- IntArrayList gift::sub2drugList
- IntArrayList gift::protein2domainList
- IntArrayList gift::domain2proteinList
- numericMatrix gift::drugSub2proteinSubMatrix
- numericMatrix gift::observedDrug2ProteinMatrix
- numericMatrix gift::vardrugSub2proteinSubMatrix
- std::vector< double > gift::loglikelyArray
- name2IndexHash gift::drugName2Index
- name2IndexHash gift::proteinName2Index
- nameList gift::drugNameList
- nameList gift::proteinNameList
- nameList gift::drugSubNameList
- nameList gift::proteinSubNameList
- nameList gift::predictDrugNameList
- nameList gift::predictProteinNameList
- nameList gift::predictDrugNameList_WithSubs
- nameList gift::predictProteinNameList_WithSubs
- IntArrayList gift::predictDrug2SubList
- IntArrayList gift::predictProtein2SubList

6.4 main.cpp File Reference

```
#include <ctime>
#include <chrono>
#include <iostream>
#include <boost/program_options.hpp>
#include "gift.hpp"
```

Functions

- int main (int argc, char ∗∗argv)

6.4.1 Function Documentation

6.4.1.1 int main (int *argc,* char ∗∗ *argv*)

References gift::helpGift(), gift::outRecord(), gift::EM::outTrainResult(), gift::EM::outTrainVariance(), gift::EM↵::predictEM(), gift::parameters::task, gift::EM::trainEM(), gift::updateTime(), gift::EM::varEM(), and gift::version().

```
                                {
  if (argc < 2){
    gift::helpGift();
    return SUCCESS;
  } // end of if.
  // read program options.
  std::string configureFileName;
  // reference to: radmangames online post.
  try{
    namespace po = boost::program_options;
    po::options_description desc("Options");
    desc.add_options()
      ("help,h","Print help messages.")
      ("version,v","Print version information.")
      ("configure,c",po::value<std::string>(&configureFileName)->required(),
       "Read the configure file.");
    po::variables_map vm;
    try{
      po::store(po::parse_command_line(argc, argv, desc), vm);
      if(vm.count("help") || vm.count("-h")) {
        gift::helpGift();
      } // end of if
      if(vm.count("version") || vm.count("-v")) {
        std::cout<<"GIFT VERSION: "<<gift::version<<std::endl;
        std::cout<<"UPDATE TIME: "<<gift::updateTime<<std::endl;
      } // end of if

      po::notify(vm); // throw an error if there are any problems.

    } catch(po::error& e){
      std::cerr<<"ERROR: " <<e.what()<<std::endl<<std::endl;
      std::cerr<< desc <<std::endl;
      return ERROR_IN_COMMAND_LINE;
    } // end of catch
  } catch(std::exception& e){
    std::cerr<< "Unhandled  Exception reached the top of main: "
             <<e.what() <<", gift will now exit."<<std::endl;
    return ERROR_UNHANDLED_EXCEPTION;
  } // end of catch

  // run program based on task.
  try{
    std::cout<<"Now start running gift..."<<std::endl;
    std::chrono::system_clock::time_point timeS =
      std::chrono::system_clock::now();
    std::time_t PtimeS = std::chrono::system_clock::to_time_t(timeS);
    std::cout<<"Job start at "<<std::ctime(&PtimeS)<<std::endl;

    gift::parameters getParameters(configureFileName);
    gift::EM EMgiftor(getParameters);
```

```
      if (getParameters.task.compare("predict") == 0 ) {
        // check is it enough?
        EMgiftor.predictEM();
        gift::outRecord(getParameters);
      } else if (getParameters.task.compare("train") == 0){
        // check is it enough?
        EMgiftor.trainEM();
        EMgiftor.varEM();
        // out train result.
        gift::outRecord(getParameters);
        EMgiftor.outTrainResult();
        EMgiftor.outTrainVariance();
      } else {
        std::cout<<"NO TASK IS SPECIFIED."<<std::endl;
        return ERROR_IN_TASK;
      } // end of if else if else.
    } catch (const std::string & e){
      std::cerr<<"ERROR: "<<e<<std::endl;
      return ERROR_IN_READFILE;
    }
    return SUCCESS;
  } // end of main
```

6.5 parameterGift.cpp File Reference

```
#include <boost/program_options.hpp>
#include <boost/any.hpp>
#include "gift.hpp"
```

Namespaces

- gift

Functions

- const char ∗ gift::BoolToString (bool b)

Index

~EM
    gift::EM, 20

alphaEB
    gift::parameters, 37
author
    gift, 8

betaEB
    gift::parameters, 37
BoolToString
    gift, 8

CPIsRec
    gift::parameters, 38
chemfpRec
    gift::parameters, 38
colNum
    gift::rowCol, 41

domain2proteinList
    gift, 16
domainNum
    gift::parameters, 38
drug2proteinFileName
    gift::parameters, 38
drug2proteinList
    gift, 16
drug2subFileName
    gift::parameters, 38
drug2subList
    gift, 16
drugName2Index
    gift, 16
drugNameList
    gift, 16
drugNameListFile
    gift::parameters, 38
drugNum
    gift::parameters, 38
drugSub2proteinSubFileName
    gift::parameters, 38
drugSub2proteinSubMatrix
    gift, 16
drugSubNameList
    gift, 17
drugSubNameListFile
    gift::parameters, 38

EMGift.cpp, 43
EStep
    gift::EM, 20
EStepThread
    gift::EM, 20
EM
    gift::EM, 20
email
    gift, 8

fn
    gift::parameters, 38
fp
    gift::parameters, 38
functionThread
    gift, 9
    gift::EM, 21
functionsGift.cpp, 43

getIndexFromHash
    gift, 9
gift, 7
    author, 8
    BoolToString, 8
    domain2proteinList, 16
    drug2proteinList, 16
    drug2subList, 16
    drugName2Index, 16
    drugNameList, 16
    drugSub2proteinSubMatrix, 16
    drugSubNameList, 17
    email, 8
    functionThread, 9
    getIndexFromHash, 9
    helpGift, 9
    IntArrayList, 8
    IntList, 8
    loglikelyArray, 17
    Matrix2Fingerpints, 10
    Matrix2FingerprintsByColumn, 11
    name2IndexHash, 8
    nameList, 8
    numericMatrix, 8
    observedDrug2ProteinMatrix, 17
    outRecord, 12
    predictDrug2SubList, 17
    predictDrugNameList, 17
    predictDrugNameList_WithSubs, 17
    predictProtein2SubList, 17
    predictProteinNameList, 17
    predictProteinNameList_WithSubs, 17
    printIntArrayList, 12

printMatrix, 13
protein2domainList, 17
proteinName2Index, 18
proteinNameList, 18
proteinSubNameList, 18
readMatrix, 13
readName2IndexHash, 13
readNameListFromFile, 14
readNameMatrixFromFile, 14
recLogLeastNum, 18
rowColFile, 15
sub2drugList, 18
updateTime, 16
vardrugSub2proteinSubMatrix, 18
version, 16
writeMatrix, 16
gift.hpp, 44
gift::EM, 19
∼EM, 20
EStep, 20
EStepThread, 20
EM, 20
functionThread, 21
iterdrugSub2ProteinSub, 21
MStep, 21
MStepThread, 22
outTrainResult, 22
outTrainVariance, 22
predictDrugs, 23
predictDrugsProteins, 23
predictDrugsProteinsWithSubs, 24
predictDrugsWithSubs, 25
predictDrugsWithSubsProteins, 25
predictDrugsWithSubsProteinsWithSubs, 26
predictEM, 27
predictProteins, 27
predictProteinsWithSubs, 28
recLoglikely, 29
setLoglikely, 29
setPointerDrug2Protein, 29
setPointerDrug2Sub, 29
setPointerDrugSub2ProteinSub, 29
setPointerProtein2Sub, 30
trainEM, 30
varEM, 30
gift::parameters, 31
alphaEB, 37
betaEB, 37
CPIsRec, 38
chemfpRec, 38
domainNum, 38
drug2proteinFileName, 38
drug2subFileName, 38
drugNameListFile, 38
drugNum, 38
drugSub2proteinSubFileName, 38
drugSubNameListFile, 38
fn, 38
fp, 38
InitDrugName2Index, 34
InitDrugSub2ProteinSub, 35
InitDrugSubNameList, 35
InitObservedDrug2ProteinMatrix, 35
InitPredictParameters, 36
InitProteinName2Index, 36
InitProteinSubNameList, 36
InitVarDrugSub2ProteinSub, 37
inputDelims, 38
iterNum, 38
loglikelyRecord, 39
outDrugSub2ProteinSubFileName, 39
outPredictCPIsFileName, 39
outRecordFileName, 39
outVarDrugSub2proteinSubFileName, 39
outputDelims, 39
parameters, 32
predictDrugsFileName, 39
predictDrugsFileName_WithSubs, 39
predictProteinsFileName, 39
predictProteinsFileName_WithSubs, 39
protein2subFileName, 39
proteinNameListFile, 39
proteinNum, 39
proteinSubNameListFile, 40
proteinfpRec, 39
setDomainNum, 37
setDrugNum, 37
setProteinNum, 37
setSubNum, 37
subNum, 40
task, 40
thread, 40
gift::rowCol, 40
colNum, 41
rowCol, 40
rowNum, 41

helpGift
gift, 9

InitDrugName2Index
gift::parameters, 34
InitDrugSub2ProteinSub
gift::parameters, 35
InitDrugSubNameList
gift::parameters, 35
InitObservedDrug2ProteinMatrix
gift::parameters, 35
InitPredictParameters
gift::parameters, 36
InitProteinName2Index
gift::parameters, 36
InitProteinSubNameList
gift::parameters, 36
InitVarDrugSub2ProteinSub
gift::parameters, 37
inputDelims

gift::parameters, 38
IntArrayList
gift, 8
IntList
gift, 8
iterNum
gift::parameters, 38
iterdrugSub2ProteinSub
gift::EM, 21

loglikelyArray
gift, 17
loglikelyRecord
gift::parameters, 39

MStep
gift::EM, 21
MStepThread
gift::EM, 22
main
main.cpp, 46
main.cpp, 46
main, 46
Matrix2Fingerpints
gift, 10
Matrix2FingerprintsByColumn
gift, 11

name2IndexHash
gift, 8
nameList
gift, 8
numericMatrix
gift, 8

observedDrug2ProteinMatrix
gift, 17
outDrugSub2ProteinSubFileName
gift::parameters, 39
outPredictCPIsFileName
gift::parameters, 39
outRecord
gift, 12
outRecordFileName
gift::parameters, 39
outTrainResult
gift::EM, 22
outTrainVariance
gift::EM, 22
outVarDrugSub2proteinSubFileName
gift::parameters, 39
outputDelims
gift::parameters, 39

parameterGift.cpp, 47
parameters
gift::parameters, 32
predictDrug2SubList
gift, 17
predictDrugNameList
gift, 17
predictDrugNameList_WithSubs
gift, 17
predictDrugs
gift::EM, 23
predictDrugsFileName
gift::parameters, 39
predictDrugsFileName_WithSubs
gift::parameters, 39
predictDrugsProteins
gift::EM, 23
predictDrugsProteinsWithSubs
gift::EM, 24
predictDrugsWithSubs
gift::EM, 25
predictDrugsWithSubsProteins
gift::EM, 25
predictDrugsWithSubsProteinsWithSubs
gift::EM, 26
predictEM
gift::EM, 27
predictProtein2SubList
gift, 17
predictProteinNameList
gift, 17
predictProteinNameList_WithSubs
gift, 17
predictProteins
gift::EM, 27
predictProteinsFileName
gift::parameters, 39
predictProteinsFileName_WithSubs
gift::parameters, 39
predictProteinsWithSubs
gift::EM, 28
printIntArrayList
gift, 12
printMatrix
gift, 13
protein2domainList
gift, 17
protein2subFileName
gift::parameters, 39
proteinName2Index
gift, 18
proteinNameList
gift, 18
proteinNameListFile
gift::parameters, 39
proteinNum
gift::parameters, 39
proteinSubNameList
gift, 18
proteinSubNameListFile
gift::parameters, 40
proteinfpRec
gift::parameters, 39

readMatrix
gift, 13
readName2IndexHash
gift, 13
readNameListFromFile
gift, 14
readNameMatrixFromFile
gift, 14
recLogLeastNum
gift, 18
recLoglikely
gift::EM, 29
rowCol
gift::rowCol, 40
rowColFile
gift, 15
rowNum
gift::rowCol, 41

setDomainNum
gift::parameters, 37
setDrugNum
gift::parameters, 37
setLoglikely
gift::EM, 29
setPointerDrug2Protein
gift::EM, 29
setPointerDrug2Sub
gift::EM, 29
setPointerDrugSub2ProteinSub
gift::EM, 29
setPointerProtein2Sub
gift::EM, 30
setProteinNum
gift::parameters, 37
setSubNum
gift::parameters, 37
sub2drugList
gift, 18
subNum
gift::parameters, 40

task
gift::parameters, 40
thread
gift::parameters, 40
trainEM
gift::EM, 30

updateTime
gift, 16

varEM
gift::EM, 30
vardrugSub2proteinSubMatrix
gift, 18
version
gift, 16

writeMatrix
gift, 16