BernoulliMix

Program package for finite mixture models of multivariate Bernoulli distributions
Edition 1.1, March 2009

Jaakko Hollmén (Jaakko.Hollmen@tkk.fi)

This is the documentation for BernoulliMix, a program package for working with finite mixture models of multivariate Bernoulli distributions. The current documentation is Edition 1.1, last updated 27 March 2009, of BernoulliMix program package, version 1.1. For the newest version available, check the version information on BernoulliMix homepage at `http://www.cis.hut.fi/jhollmen/BernoulliMix`.

Send bug reports and suggestions to `Jaakko.Hollmen@tkk.fi`.

Table of Contents

1 Introduction

Binary data sets arise in many practical applications as categorical indicator variables denoting dichotomies such as sick vs. healthy (or even worse: dead vs. alive), positive vs. negative, defective vs. non-defective, success vs. failure, present vs. absent among others. Even whole databases are represented using this categorical representation, for instance supermarket basket data, computer and telecommunications systems data, text document data, and the like. Binary data, or 0-1 data, may arise as a natural way to represent the measured variables, or as a transformed representation of the original variable through quantization or other form of abstraction. In machine learning (Bishop, 2006) and data mining (Hand, Mannila, Smyth, 2001), researchers have been interested in modeling 0-1 data from many perspectives, including local patterns such as frequent itemsets (Hand, Mannila, Smyth, 2001), global models in a probabilistic context (Tikka and Hollmén, 2007), and a combination of global and local approaches (Hollmén, Seppänen, Mannila, 2003). Large-scale bioinformatics application involving a database of 0-1 data has been reported in (Myllykangas et al., 2008). Whereas common machine learning and data mining books present mixture models on a general level, they are covered in more detail, for instance, in (Wolfe, 1970; Titterington et al. 1985; McLachlan and Basford, 1987; McLachlan, 1996; McLachlan 2000).

The BernoulliMix program package approaches modeling of 0-1 data in a probabilistic framework using finite mixture models of multivariate Bernoulli distributions. The target audience for BernoulliMix program package includes researchers, teachers, and students in the fields of machine learning and data mining. Some exercises are included in the documentation in the hope that they are useful for educational purposes on machine learning and data mining courses. They have been used on the machine learning courses of the author to form the term project. We have reported our experiences in using a ready-to-use program package in machine learning education in (Hollmén and Raiko, 2008). Instead of "just getting their own programs to work", students concentrate on setting up the experiments, and thinking about the results. An important note is that this documentation is merely a description of the BernoulliMix program package and should not be considered as teaching material on finite mixture models of multivariate Bernoulli distributions nor learning from data in that context. At best, students learn the concepts of mixture models and learning from data in the classroom and by using the BernoulliMix program package, they will see the classroom concepts come to live in action!

The BernoulliMix program package contains five programs to work with finite mixture models of multivariate Bernoulli distributions. With BernoulliMix, users can

1. Initialize the mixture model with randomly selected parameters (`bmix_init`)
2. Calculate the likelihood of data with the mixture model (`bmix_like`)
3. Train the mixture model from data using the EM algorithm (`bmix_train`)
4. Sample data from the mixture model (`bmix_sample`)
5. Cluster data with the mixture model by the maximum posterior rule (`bmix_cluster`)

After describing the step towards successful installation, we describe a test suite to ensure that all the programs are running correctly. Some example 0-1 data sets are included in the package, which are described in a later chapter. In addition, the program package includes an example how to customize the package by writing additional functionality of your own.

This documentation helps you to become familiar with BernoulliMix program package. See Chapter 2 [Getting started with BernoulliMix], page 2 for the contents of the package including installation instructions and a short presentation of the test suite. See Chapter 3 [Programs in BernoulliMix program package], page 4 for the five programs to work with finite mixture models of multivariate Bernoulli distributions. See Chapter 4 [Examples of 0-1 data sets], page 11 for the descriptions of example 0-1 data sets for your use. See Chapter 5 [Extending BernoulliMix], page 12 on information how to extend the functionality of BernoulliMix program package.

2 Getting started with BernoulliMix

In this chapter, we cover how to install BernoulliMix program package on your computer by compiling its programs that are written in C programming language. Also, we present a test suite for ensuring that BernoulliMix program package is behaving expectedly. If you are reading this, the chances are high that you have already downloaded the BernoulliMix program package and familiarized yourself with the documentation. After all, this is the documentation!

2.1 Installing BernoulliMix program package

The BernoulliMix program package is distributed with the source code of all the programs in programming language C and a '`Makefile`' utility that helps the user to compile the source code into executable programs. This installation information is targeted to users with familiarity on compiling programs Linux operating systems. If your system is somehow related to UNIX or Linux, there is a high probability that you will be able to compile the programs without any problems. See the BernoulliMix home page for a list of compatible systems.

The package is contained in a file '`bmix-1.1.tar.gz`', which is a compressed archive of files, available at BernoulliMix home page. You should uncompress the gzipped tar file with the commands

```
~ % gunzip bmix-1.1.tar.gz
~ % tar xvf bmix-1.1.tar
```

After uncompressing, you will have a directory '`bmix-1.1/`' created for you. Enter the created directory '`bmix-1.1/`' with the command `cd bmix-1.1/` and you should see files '`COPYING`', '`Changes`', '`INSTALL`', '`Makefile`', and '`README`' and in addition, the directories named '`bin/`', '`doc/`', '`src/`', '`data/`', and '`test/`'. The files '`README`' and '`INSTALL`' contain brief information for the impatient users. The directory '`bin/`' is initially empty and will contain the binary files once they are compiled. Directory '`doc/`' contains the documentation in DVI format as '`bmix_doc.dvi`', in PostScript format as '`bmix_doc.ps`', and as a PDF file '`bmix_doc.pdf`'. Also, the source file for the documentation in Texinfo format is provided. The directory '`src/`' contains the source files in C programming language that you are free to inspect, learn from and modify under the terms of the GNU General Public License (in file '`COPYING`'). There is a further directory '`src/my_getopt/`' that contains a separate, command-line parser contributed by Benjamin Stiller, which has its own licensing terms (see the file '`src/my_getopt/LICENSE`'). The directory '`data/`' contains example 0-1 data sets and the directory '`test/`' contains the test suite that will be explained shortly.

In order to compile the programs to executable form, you need a compiler and compilation instructions. The compilation instructions are provided in the file '`Makefile`'. To compile the programs, write command `make` on your command prompt in a shell window, in the directory '`bmix-1.1/`' containing the '`Makefile`'. This starts the necessary compilation process and produces the programs described in this documentation. To be able to run the programs, you can add the directory with the executable programs to your environment variable `PATH`, or alternatively give the full path to the programs when running them or simply execute them in the '`bmix-1.1/`' directory.

```
~/bmix-1.1/ % /home/myaccount/bmix-1.1/bin/bmix_train --data mydata ...
~/bmix-1.1/ % ./bin/bmix_train --data mydata ...
```

After the compilation, you can test that everything works correctly by following the testing instructions in the next section.

2.2 Testing BernoulliMix program package

In order to test that you have compiled everything correctly and that the BernoulliMix program package is working expectedly, you can go to the directory '`test/`' and run the test suite. The

test suite has been written as a Korn shell script that you can execute with the command `ksh bmix_test` (or just `./bmix_test`). The test suite runs all five programs in a typical work flow that feeds data and models to next programs. The test suite should not create any errors. The test suite monitors the program behavior by storing their return status and interpreting the overall results in the end. If error occurs during the test, the program reports that there are errors and aborts prematurely. An example printout of a test run that runs correctly without any errors looks like

```
~/bmix-1.1/test %
~/bmix-1.1/test % ksh bmix_test

Test script bmix_test: testing the BernoulliMix program package

--- Started bmix_test ---
Testing BernoulliMix, mixture model: c = 5, data: n = 1000, d = 2, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 5000, d = 2, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 1000, d = 5, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 5000, d = 5, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 1000, d = 10, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 5000, d = 10, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 1000, d = 20, ok
Testing BernoulliMix, mixture model: c = 5, data: n = 5000, d = 20, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 1000, d = 2, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 5000, d = 2, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 1000, d = 5, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 5000, d = 5, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 1000, d = 10, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 5000, d = 10, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 1000, d = 20, ok
Testing BernoulliMix, mixture model: c = 10, data: n = 5000, d = 20, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 1000, d = 2, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 5000, d = 2, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 1000, d = 5, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 5000, d = 5, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 1000, d = 10, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 5000, d = 10, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 1000, d = 20, ok
Testing BernoulliMix, mixture model: c = 20, data: n = 5000, d = 20, ok
--- Ended bmix_test ---

Called BernoulliMix programs 432 times, all tests ok.

~/bmix-1.1/test %
~/bmix-1.1/test %
```

All is well that ends well, indicated by `all tests ok.`

Exercises

1. On what computer and operating system did you compile the BernoulliMix program package? The author of the package would be very happy if you sent the printout of your test results by e-mail. The list of compatible platforms will be listed on BernoulliMix home page. To find out what system you are using, type the command `uname -a` in a shell window.

3 Programs in BernoulliMix program package

There are five programs in BernoulliMix program package to work with finite mixture models of multivariate Bernoulli distributions. With BernoulliMix, users can

1. Initialize the mixture model with randomly selected parameters (`bmix_init`)
2. Calculate the likelihood of data with the mixture model (`bmix_like`)
3. Train the mixture model from data using the EM algorithm (`bmix_train`)
4. Sample data from the mixture model according to ancestral sampling scheme (`bmix_sample`)
5. Cluster data with the mixture model by the maximum posterior rule (`bmix_cluster`)

All the programs are run from the command-line and can be combined in a flexible fashion, for instance, to initialize and learn models from data. All inputs and outputs of the programs are controlled with command-line options. There are alternative forms of command line options: you may specify an option with a short format, such as '`-f`' followed by the argument, or in longer format, such as '`--data`'. The longer format is much more readable and less prone to mistakes in practice. The argument to the option follows the option immediately or with a space between the option and the argument. Options may be given in any order, but the argument to the option (naturally) always follows the option. A space is recommended for improved readability, since the mistaken form '`-model-in`' really means '`-m odel-in`', that tells to write the model to a file '`odel-in`'. The correct form would be '`--model-in my.model`'. See following sections for examples.

The five programs making the BernoulliMix program package are described in the following sections. The basic function of each program and the available command-line options are described. This basic usage is followed by examples to demonstrate practical use scenarios and exercises to be used for educational purposes. The programs explained in the following sections are `bmix_init`, `bmix_train`, `bmix_like`, `bmix_sample`, and `bmix_cluster`.

3.1 Initialize the mixture model parameters with `bmix_init`

The program '`bmix_init`' initializes a mixture model with random parameter values and outputs the initialized model. If a filename is specified as an argument to the option '`-o`', the model is written to a file, otherwise the model is printed on the screen (or technically speaking, the standard output). The screen output can, of course, be redirected to a file, for instance.

With '`bmix_init`', the mixing coefficients are always initialized to be equal and they will sum to one. The parameters in the component distributions are drawn randomly from a uniform distribution between desired probability values, or if not specified, between 0.25 and 0.75.

The program '`bmix_init`' accepts the following options:

'`--data-dim,-d`'

The dimension of the data must be specified with the option '`--data-dim`' or with the corresponding short form '`-d`'. Dimension of the data is necessary to specify the model; this option is therefore mandatory.

'`--clusters, -c`'

The number of component distributions are given with the option '`--clusters`', which has the short form '`-c`'. Also, the number of component distributions (or clusters in the clustering context) is necessary to specify the model; this option is mandatory.

'`--model-out, -o`'

Optionally, a model filename can be specified with the long option '`--model-out`' or its short form '`-o`'. It will write over any file without asking, so care is needed. If the option '`--model-out`' is not given, the initialized model will be printed on the screen. An error is given if it is not possible to open the model file for writing purposes.

‘`--min-probability, -a`’

This option is used to give the minimum value for random parameters of the component distributions. Its use is optional. You can alternatively use the long version ‘`--min-probability`’ or the short version ‘`-a`’. An error is given if the probability value given as an argument is beyond the natural range between 0 and 1, or if the minimum probability is larger than the maximum probability. The default value is 0.25. It is technically possible to give 0.0 as the minimum value, but this may lead to unwanted results (see Exercises).

‘`--max-probability, -b`’

This option specifies the maximum value for random parameters of the component distributions. Its use is optional. Long version of the option is ‘`--max-probability`’ and the short version is ‘`-b`’. An error is given if the probability value is beyond the natural range between 0 and 1, or if the maximum probability is smaller than the value for minimum probability. The default value is 0.75.

‘`--help, -h`’

The option ‘`--help`’ prints out the available options for the program, both in short and long forms, with their short description.

Examples

The first example demonstrates a useful feature present in all programs. Try it out and see for yourself:

```
./bmix_init --help
```

The following example command initializes a mixture model with 2 component distributions for data with data dimension 3. The parameters for the component distributions are drawn from a uniform distribution between 0.25 and 0.75 (which is the default). The model is written to a model file ‘`small.model`’.

```
./bmix_init --data-dim 3 --clusters 2 --model-out small.model
```

Same command as the previous one can also be given using the short options. Instead of writing the mixture model to a file, the model is printed on screen.

```
./bmix_init -d 3 -c 2
```

The function of the first example may also be achieved using the shell redirection with the command

```
./bmix_init --data-dim 3 --clusters 2 > small.model
```

The next example initializes a mixture model with 6 component distributions. The dimension of data is 4. The parameters are drawn for a uniform distributions between 0.2 and 0.8. The model is printed on screen.

```
./bmix_init -d 4 -c 6 --min-probability 0.2 --max-probability 0.8
```

Exercises

1. Create a mixture model with 3 component distributions for modeling data with data dimension 4. Identify the parameters of the mixture model in the resulting model file, that is, find the correspondence of the mixture model equation for calculating and the numbers on the model file.
2. Explain why the following initialization of the mixture model isn't so useful (or even sensible) in practice?

   ```
   ./bmix_init --data-dim 3 --clusters 4 --min-probability 0.5 \
       --max-probability 0.5
   ```
3. What kind of consequences does it have if some of the parameters of the component distributions are initialized with zero values? Think in terms of the update equation of the mixture model.

3.2 Train the mixture model parameters with `bmix_train`

The program '`bmix_train`' trains a finite mixture model of multivariate Bernoulli distributions using the Expectation-Maximization (EM) algorithm. In order to specify the model, you have to define the number of component distributions and the dimension of the data. In this case, the model parameters will automatically be initialized randomly. Alternatively, you may give an existing model as an input to the program. As parameters for the iterative EM training procedure, '`bmix_train`' takes the maximum number of iterations and/or the relative tolerance for the change in the likelihood as a stopping criterion. If both are given, the training terminates when either the maximum number of iterations is reached, or when the relative change in the likelihood is smaller than the tolerance given as an argument to the option. Equivalent sample size affects the estimation of the mixing coefficients by inputting virtual data points (pseudo data) to the mixture components. When running the training procedure, the current log-likelihood value is printed for each iteration. If no output is wished during the execution of the programs, so called quiet mode can be invoked.

The program '`bmix_train`' accepts the following options:

'`--data, -f`'

The filename of the data set used during training the mixture model is given as an argument. This option is mandatory.

'`--model-in, -i`'

The filename of the initial model used in training is given as an argument to this option. If this option is not used, the mixture model is automatically initialized with random parameter values using the default settings.

'`--model-out, -o`'

The filename of the final model is given as an argument to this option. The model output is always written to a file. This is a mandatory option.

'`--clusters, -c`'

The number of component distributions used in the model. This command-line option is only used if no initial model is given and can not be used together with the option '`--model-in`', since the model already contains this information.

'`--iterations, -t`'

The maximum number of iterations used during the training. The default is 100 iterations, which is used if this option is not given.

'`--relative-change, -r`'

The option '`--relative-change`' specifies a stopping criterion based on the relative change of likelihood between successive iterations of the EM algorithm. When the training converges, the relative change in likelihood becomes smaller and smaller. When the change is smaller than the specified tolerance, the training stops. This option can be used in connection with the maximum number of iterations.

'`--equivalent-sample-size, -e`'

Assign pseudo data, or virtual data points to the component distributions distributed equally among the component distributions.

'`--quiet, -q`'

The quiet mode suppresses any printing.

'`--help, -h`'

Print a help indicating the options available in the '`bmix_train`'.

Examples

The first example initializes and trains a mixture model with 2 component distributions using the data set '`marker.data`' with data dimension 6. After training for 10 iterations, the trained model is written to a file '`cancer.model`'.

```
./bmix_train --data marker.data --data-dim 6 --clusters 2 \
     --iterations 10 --model-out cancer.model
```

The second example reads an existing model file '`init.model`' and trains a mixture model with a data set '`stuff.data`' for a very large number of iterations, or until the relative change of likelihood is less than 0.005 (whichever comes first, most probably the relative change). The trained model is written to a model file '`final.model`'. Neither the dimension of the data nor the number of components of the mixture model need to be specified, since the initial model includes this information.

```
./bmix_train --data stuff.data --model-in init.model \
     --iterations 10000 --relative-change 0.005 --model-out final.model
```

Exercises

1. As an exercise, write the command in the first example using the short options.
2. Initialize a mixture model with two component distributions to model data with data dimension six and write the file to a model file called '`init.model`'. Then, using the same initial model '`init.model`' and the same data set '`marker.data`', train a model three separate times and write each of the resulting models to a file, say '`out1.model`', '`out2.model`', '`out3.model`'. For training, use commands like the following

   ```
   ./bmix_train --model-in init.model -f marker.data -t 20 \
        --model-out out1.model
   ```

   Compare the resulting models, and explain your observations.
3. Repeat the following training command three times (with different model names, such as '`m1.model`', '`m2.model`', '`m3.model`'). Explain why the results are not identical?

   ```
   ./bmix_train -c 2 --data marker.data --data-dim 6 -t 20 \
        --model-out m1.model
   ```

4. Run one iteration of the EM algorithm for finite mixture model of multivariate Bernoulli distributions with pen and paper (in the style of course exercise). First, calculate the posterior probabilities of each data vector in each component distribution (the E-step), and then perform the update of the parameters (M-step). Use a data set with two data vectors of dimension two shown as follows.

   ```
   1 1
   0 1
   ```

   Use the following model below as your initial model.

   ```
   2 2
   # A finite mixture model with c=2, d=2
   # The mixture coefficients:
   0.7 0.3
   # The paramaters of the component distributions:
   0.5 0.4
   0.2 0.7
   ```

   Verify your results by running one iteration of EM with BernoulliMix. Make sure you get the same results.

3.3 Calculate the likelihood of data with the mixture model with `bmix_like`

The program `bmix_like` calculates the probability or likelihood of data given the model. The likelihood is the logarithm of the probability, either separately for individual data vectors or for the whole data set as one number. Results are always printed on screen, shell redirection can be used to write results to a file.

The program `bmix_like` accepts the following options:

'`--data, -f`'

The name of the file following this option specifies the data set used in likelihood calculation.

'`--model-in, -i`'

The name of the file following this option specifies the mixture model used in likelihood calculation.

'`--sample-likelihood, -s`'

If the option '`-s`' is used, the program outputs likelihoods for every data vector, or sample, separately. The output is printed on screen with one likelihood for each row (data vector) in the data set.

'`--total-likelihood, -l`'

With the option '`-l`', the program calculates the total likelihood for the whole data set assuming independence of data vectors in the data set. Average likelihood per data sample is given and printed on the screen. This is the default, if neither of the options '`-s`' or '`-l`' are given.

Examples

The first example reads the data from a file '`small.data`' and a model from a file '`tiny.model`' and prints the average likelihood per sample for the whole dataset on screen.

```
./bmix_like --data small.data --model-out tiny.model --total-likelihood
```

The second example reads the data from a file '`this.data`' and a model from a file '`that.model`' and prints the likelihood per sample on the screen, on as many rows as there are rows in the data file '`this.data`'.

```
./bmix_like -f this.data -i that.model --sample-likelihood
```

The third example is a repetition of the second example, now executed with short options. Results are redirected to a file '`likelihoods.txt`'.

```
./bmix_like -f this.data -i that.model -l > likelihoods.txt
```

Exercises

1. Initialize 10 separate mixture models with two component distributions to work with the data set '`marker.data`' (the dimension of the data set is six). Calculate the total likelihood of the data '`marker.data`' with each of the initial models and store the values. Then, train each of the mixture models until convergence and calculate the total likelihood of the data '`marker.data`' in the same way as above, but now for the trained models. Compare the two sets of the likelihood values (10 values each) and explain your observations. You can do the comparison visually with a boxplot, for instance.

3.4 Sample data from the mixture model with `bmix_sample`

The program '`bmix_sample`' samples data from the mixture model using the ancestral sampling scheme (Bishop, 2006). The sampled data is printed on the screen. If you want the sampled data written to a file, you must use shell redirection (see Examples). The mandatory options to the program are the name of the model file given as an argument to the option '`-i`' and the number of samples as an argument to the option '`-n`'.

The program '`bmix_sample`' accepts the following options:

'`--model-in, -i`'

The name of the file given as an argument to this option specifies the mixture model to sample from. This option is mandatory.

'`--number-of-samples, -n`'

The number of sampled data vectors is given as an argument to this option. This option is mandatory.

'`--help, -h`'

The option '`--help`' prints out the available options for the program, both in short and long forms, with their short description.

Examples

The following example reads the model '`tiny.model`' and samples 1000 data vectors according to the mixture model. The command in the example prints the sampled data on the screen.

```
./bmix_sample --model-in tiny.model --number-of-samples 1000
```

This example differs from the previous example in that the sampled data is written to a output data file '`samples.data`'.

```
./bmix_sample --model-in tiny.model --number-of-samples 1000 > samples.data
```

Exercises

1. Train a mixture model with 5 component distributions using the data set '`dna_17.data`' until convergence and sample 100, 1000, 10000 data vectors from the trained model. Save the sampled data to separate data files. Train three model, one from each of the sampled data set and compare the resulting three models with the original trained model from which you generated the samples. Think also how you can actually compare two models? Think of solutions in terms of likelihood calculations.

3.5 Cluster data with the mixture model with `bmix_cluster`

The program '`bmix_cluster`' clusters the data set into clusters according to the posterior probabilities of the component distributions. A data vector is clustered to the component distribution that maximizes its posterior probability. Clustering with '`bmix_cluster`' is a hard clustering, or partitioning of the data set into disjoint subsets. As an output, one can choose to print all data belonging to one cluster (defined by a pre-defined component distribution), or to print the cluster indices of all the data vectors. Printing the cluster indices is the default behavior, unless the option '`--cluster`' is given. The number of a cluster is defined from 1 to J in the order they occur in the model file. The results are always printed on screen, but can be written to a file using shell redirection.

The program '`bmix_cluster`' accepts the following options:

'`--data, -f`'

The data file following the option '`--data`' is the data set to be clustered. This option is mandatory.

'`--model-in, -i`'

The argument used with the option '`--model-in`' determines the model used in clustering, that is, in calculating the posterior probabilities of the component distributions and allocating data to the component distribution that has the maximum posterior probability. This option is mandatory.

'`--cluster, -c`'

You can specify the number of a cluster as an argument to the option '`--cluster`'. In this case, the printed data will have the maximum posterior probability in the given component distribution among all of the component distributions. The number of a cluster ranges from 1 to the total number of component distributions (defined from 1 to J in the order they occur in the model file). Giving this option overrides the default behavior of printing out the cluster indices for each data vector separately.

'`--help, -h`'

Prints the help and the available options to the program.

Examples

The first example takes the model '`tiny.model`' and calculates the posterior probabilities of the component distributions for each data vector in the data file '`my.data`' and prints out the data belonging to the cluster 1 according to the maximum posterior probability rule. The first cluster is defined by the first component distribution in the model file.

```
./bmix_cluster --model-in tiny.model --data my.data --cluster 1
```

The second example performs the same clustering as the first example, but prints the cluster indices on the screen.

```
./bmix_cluster --model-in tiny.model --data my.data
```

Exercises

1. Perform a clustering as presented in the first example above with a trained model and compare the average of the clustered data with the corresponding parameter vector of the component distribution.
2. Model selection refers to selecting the number of component distributions in a mixture model. Repeat the model selection procedure presented in (Tikka and Hollmén, 2007) for the included data set '`dna_amp_chr_17.data`' (For a more detailed description, see the next chapter). Try out different solutions ranging from 2 clusters to 30 clusters in a cross-validation setting and base your selection on the average likelihood (averaged over repeated runs). For cross-validation, you need to be able to divide data sets to a training part and a validation part and store the results of the repeated runs of the training procedure. Plot the results similarly to the results of (Tikka and Hollmén, 2007). How many components would you select?

4 Examples of 0-1 data sets

In this chapter, we describe the format of the data and a few 0-1 data sets that are included as examples in the BernoulliMix program package. The example data sets are located in the directory '`bmix-1.1/data`'. In addition to the example data sets, you can inspect the BernoulliMix home page at the address `http://www.cis.hut.fi/jhollmen/BernoulliMix` for more pointers on binary (0-1) data matrices. You are also welcome to submit your own.

The data files are text files that contain data vectors as rows of 0's and 1's separated by whitespace on each row, the rows ended with a newline. The lines are concatenated together to form the data set as shown in the following.

```
[0|1]<whitespace>[0|1]<whitespace>...[0|1]<newline>
[0|1]<whitespace>[0|1]<whitespace>...[0|1]<newline>
...
[0|1]<whitespace>[0|1]<whitespace>...[0|1]<newline>
```

4.1 Genetic marker data

The first data set in the file '`marker.data`' contains measurements of 6 genetic biomarkers in 38 patients. The patients are the rows in the data file, the six measurements are recorded in the columns of the data matrix. A 0 indicates a neutral result from the test, a 1 denotes an interesting or an abnormal finding. This data file serves as an example of a small illustrative data set. The first few lines of the data file are shown.

```
0 0 0 1 1 0
0 0 0 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 1 0 0 0
1 1 0 0 0 0
```

4.2 DNA copy number amplifications in chromosome 17

The second data set '`dna_amp_chr_17.dat`' contains data about certain type of mutations of DNA in cancer patients. One row of data depicts structural aberrations of the chromosome 17 in a given cancer patient. The DNA copy number amplification means that the DNA material is mutated and copied so that the DNA consists of multiple copies of the chromosomal material. The data set contains DNA amplifications of the chromosome 17 in 342 cancer patients. There are 12 chromosomal regions covering the chromosome 17 that make up the 12 attributes in each row vector. These are, in fact, recordings from the chromosomal regions with names 17p13, 17p12, 17p11.2, 17p11.1, 17q11.1, 17q11.2, 17q12, 17q21, 17q22, 17q23, 17q24, 17q25. The chromosomal regions which are amplified, are marked with ones, and the rest are marked with zeroes. For more information, see (Myllykangas et al., 2008). As you will see, there are strong correlations between adjacent attributes, since they are nearby regions in the chromosome and since amplifications can cover large areas of the chromosome. This data set is well suited for clustering. The first few lines of the data file are shown below.

```
0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 1 1 1 1
0 0 0 0 0 0 0 0 0 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 1 0 0 0 0
```

5 Extending BernoulliMix

This chapter describes how to extend BernoulliMix program package with your own programmed functions in C programming language. First and foremost, BernoulliMix is meant to be a user-level program package for working with finite mixture models of multivariate Bernoulli distributions. It is expected that most users are satisfied with the available functionality and do not have the need to extend BernoulliMix by additional programming. Therefore, the documentation will never be a full-blown developer's guide. Rather, it gives some important hints about data structures that are central to the operation of BernoulliMix program package. The data structures governing data and models are covered in the following two sections. In the last section, how to extend the BernoulliMix program package is presented by a way of example. A full application '`bmix_custom`' containing customized code to calculate the average of a 0-1 data set is presented. By following the example and learning about central data structures, extending should be relatively straightforward. It still requires familiarity working with the programming language C and the willingness (and patience) to browse through the existing program code in BernoulliMix. Before going to the example '`bmix_custom`', central data structures in BernoulliMix are presented in the next sections.

5.1 Internal representation of the data sets

Externally, the data sets in BernoulliMix program package are represented as text files containing 0's and 1's. In real-world data sets, however, the data is typically very sparse, meaning that only a very small portion of all variables are in fact 1's. The proportion of 1's in the data typically ranges from 1 percent to 5 percent. We may take advantage of this property and represent the data internally coding only the variables with 1's and treating 0's as default values. This can be achieved by storing only the indices of 1's of the data vector. As an example, a high-dimensional data vector '`0 1 1 0 0 0 0 0 0 1 0 1`' is converted to '`2 3 10 12`', since only the 2nd, 3rd, 10th, and 12th component in the data vector are 1's. The memory savings are related to the degree of sparsity in the data set.

Inside the BernoulliMix, the data vectors are stored in a linked list of `struct bmix_entry` structures. A definition of `struct bmix_entry` structure is shown below. The variable `data_dim` defines the true dimension of the original data and the needed number of index variables for storing the data in stored in `ncomp`. The pointer `comps` points to the data vector, which is coded as integer indices of 1's, as explained above. Variables `weight` and `mask` are not currently in use but may have natural use in extending the package. The field `next` points to the next data vector in the uni-directional linked list of `struct bmix_entry` structures. The `next` variable of the last vector of data has a value `NULL`.

```
/* data structure for one data vector: */
struct bmix_entry {
  unsigned int data_dim;
  unsigned int ncomp;
  unsigned int *comps;
  double *weight;
  char *mask;
  struct bmix_entry *next;
  };
```

The customized application `bmix_custom` traverses through the data set, which serves as a good example to learn from.

5.2 Internal representation of the models

In the previous section, the data structure in C programming language for representing data vectors was presented. In similar fashion, structured data is used also for multivariate Bernoulli distributions and for finite mixture models of multivariate Bernoulli distributions. A multivariate Bernoulli distribution is defined with the following data structure `struct bernoulli_entry`:

```
/* data structure for a bernoulli distribution: */
struct bernoulli_entry {
  int ncomp;
  double *params;
  char *mask;
  };
```

The data structure includes information about the number of parameters in the Bernoulli distributions in the variable `ncomp`. The pointer `params` points to an array of parameters. Currently, `mask` is not used, but is reserved for masking off variables in future use.

The structure `struct bernoulli_mixture` defines the data structure for a mixture model of Bernoulli distributions. A finite mixture model consists of mixture coefficients and component distributions. The previous data structure `struct bernoulli_entry` is used for each of the component distributions and a finite mixture model has an array containing pointers to these data structures stored in the `comp`. The mixture weights are stored in an array `mix_weight`. The variable `ncomp` tells how many component distributions there are in a mixture model. With the variable `equivalent_sample_size`, pseudo-counts can be added to the data during model training.

```
/* data structure for a mixture of bernoulli distributions: */
struct bernoulli_mixture {
int ncomp;
double *mix_weight;
double *equivalent_sample_size;
struct bernoulli_entry **comp;
};
```

During the Expectation-Maximization (EM) algorithm, training proceeds with a repeated computation of the posterior probability matrix with the help of the Bayes's theorem. This computation is in fact the E-step of the EM algorithm. The M-step of the EM algorithm takes both the data set and the posterior probability matrix as an input and produces a new set of model parameters that are used in the next iteration of the EM algorithm. The posterior probabilities only make sense in the context of a given model and a data set. Therefore, we define a data structure `struct b_likelihood` that includes pointers to the mixture model in the variable `mixt` (explained above) and a data set given by a pointer to the first data entry in the data set as `header`. The variable `table` stores the posterior probability values. The variable `loglike` stores the log-likelihood which is produced as a side-product in computing the posterior probability matrix.

```
/* Data structure for likelihood */
struct b_likelihood {
double **table;
  struct bernoulli_mixture *mixt;
  struct bmix_entry *header;
  double loglike;
};
```

In the next section, we'll go through `bmix_custom`, an example application to extend the functionality of BernoulliMix.

5.3 Extend BernoulliMix package by following the example `bmix_custom`

It is possible to create your own application programs using the data structures and existing functionality in the BernoulliMix program package. You can use any function part of BernoulliMix and combine them with your program code (under the terms of the GNU General Public Licence). As an example, there is a program template to compile a program '`bmix_custom`'. The source code of the program is in the file '`src/main_custom.c`'. The extension, programmed as an example, calculates the average value of a 0-1 data set. This functionality is not included in BernoulliMix program package as a built-in feature, but by following the example, you can learn to take advantage of the existing code and extend it for added functionality. The rules for compiling '`bmix_custom`' are included in the file '`Makefile`'. When creating a program with your own custom functionality, similar to the example '`bmix_custom`', you need to compile the core functionality in the file '`BernoulliMix.c`' and link it together with your own functionality in the file '`bmix_custom.c`'. Naturally, all files need to be compiled to object files and then linked together.

If your contributed program code extends over a screenful of code, say 20 or 30 lines of code, it is advisable to place it in another file such as '`bmix_contrib.c`' and just write the necessary function calls in '`bmix_custom.c`'. This improves the readability of your code. The example rule for compiling bmix_custom follows this model: place all your changes in '`bmix_contrib.c`' and change the '`bmix_custom.c`' minimally just to achieve the behavior you want.

If you think your contribution is useful to others and you are willing to share it, you could propose it as a contribution to the BernoulliMix package. In order to contact the author of BernoulliMix and propose a contribution, take a look at the BernoulliMix home page at `http://www.cis.hut.fi/jhollmen/BernoulliMix` for more precise information.

Acknowledgments

I wish to thank all people that have contributed to the BernoulliMix program package. Paul Grouchy was the first user of the package, he has made many useful remarks and suggestions for improvements. Niko Vuokko has actively contributed more suggestions and improvements that I have been able to incorporate. I have received helpful comments from Gemma Garriga, Heikki Mannila, Mika Sulkava, and Nikolaj Tatti concerning the programs and the documentation. Janne Toivola and Mikko Korpela helped with compatibility testing. All the students of 2008 on the course *T-61.5140 Machine Learning: Advanced Probabilistic Methods* at the Helsinki University of Technology have contributed to the current version of the package.

References

Christopher M. Bishop. **Pattern recognition and machine learning**, Springer-Verlag, 2006.

David Hand, Heikki Mannila, Padhraic Smyth. **Principles of Data Mining**, MIT Press, *Adaptive Computation and Machine Learning Series*, 2001.

Jaakko Hollmén, Jouni K. Seppänen, and Heikki Mannila. **Mixture models and frequent sets: combining global and local methods for 0-1 data**. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining*, pages 289–293. Society of Industrial and Applied Mathematics, 2003.

Jaakko Hollmén and Jarkko Tikka. **Compact and understandable descriptions of mixtures of Bernoulli distributions**. In M.R. Berthold, J. Shawe-Taylor, and N. Lavrac, editors, In *Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA 2007)*, volume 4723 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, September 2007. Ljubljana, Slovenia.

Jaakko Hollmén and Tapani Raiko. **Learning mixture models courseware for finite mixture distributions of multivariate Bernoulli distributions**. In Stéphanie Jacquemont and Colin de la Higuera, editors, *Proceedings of Teaching Machine Learning workshop on open problems and new directions*. May 2008. Saint-Étienne, France. Available on the `http://www.cis.hut.fi/jhollmen/BernoulliMix`.

Geoffrey McLachlan and David Peel. **Finite Mixture Models**. Wiley Series in Probability and Statistics. John Wiley & Sons, 2000.

Geoffrey J. McLachlan.T **The EM Algorithm and Extensions**. John Wiley & Sons, 1996.

Geoffrey J. McLachlan and Kaye E. Basford. **Mixture Models — Inference and Applications to Clustering**, volume 84 of *Statistics: Textbooks and Monographs*. Marcel Dekker, Inc., 1987.

Samuel Myllykangas, Jarkko Tikka, Tom Böhling, Sakari Knuutila and Jaakko Hollmén. **Classification of human cancers based on DNA copy number amplification modeling**. *BMC Medical Genomics*,1:15, May 2008.

Jarkko Tikka, Jaakko Hollmén, and Samuel Myllykangas. **Mixture modeling of DNA copy number amplification patterns in cancer**. In Francisco Sandoval, Alberto Prieto, Joan Cabestany, and Manuel Graa, editors, *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, volume 4507 of *Lecture Notes in Computer Science*, pages 972–979. Springer-Verlag, June 2007. San Sebastin, Spain.

D.M. Titterington, A.F.M. Smith, and U.E. Makov. **Statistical analysis of finite mixture distributions**. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1985.

John W. Wolfe. **Pattern Clustering by Multivariate Mixture Analysis**, *Multivariate Behavioral Research*, 5:329–350, July 1970.

Index

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible.

You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled

"Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified

version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.