Quick Start Manual for the MVN-HMM Toolbox

Sergey Kirshner
Donald Bren School of Information and Computer Science
University of California, Irvine

June 24, 2005

1 Overview

MultiVariate Nonhomogeneous Hidden Markov Model toolbox (MVN-HMM) is a modular collection of algorithms related to modeling sequences of vector data using hidden Markov models (HMM), commonly used to model time series or dynamical systems. The toolbox contains C++ implementations of algorithms for parameter estimation (Baum-Welch), most likely sequence estimation (Viterbi), data sampling from the model, data log-likelihood estimation as well as routines to compute simple statistics of the data, e.g. mean, correlation, etc. The model specification is quite flexible, and the models are not limited to stationary HMMs. The toolbox can train models for both real- and discrete-valued vector data although the emphasis during development was primarily on discrete-valued data.

The purpose of this manual is to allow the user to start using the toolbox with a number of commonly used models, without going into the details of all available options and modules. The manual assumes some prior knowledge of hidden Markov models, and does not provide the details of the algorithms used in the code. While the document by no means is a comprehensive manual, it should provide enough information to use the toolbox for data analysis using HMMs.

2 Brief Model Description

A hidden Markov model (HMM) is a doubly stochastic model where observations are modeled conditioned on the small number of discrete states, with Markovian transitions between the states. Let $\mathbf{R} = (\mathbf{R}_1, \ldots, \mathbf{R}_T)$ be a sequence of observations (assumed to be vectors), and let $\mathbf{S} = (S_1, \ldots, S_T)$ be the corresponding sequence of discrete (or hidden) states. The directed graphical model for an HMM is shown in Figure 1. The log-likelihood of the data under this model can be written as

$$l = \log P(\mathbf{R}) = \log \sum_{\mathbf{S}} \left[P(S_1) \prod_{t=2}^{T} P(S_t|S_{t-1}) \right] \left[\prod_{t=1}^{T} P(\mathbf{R}_t|S_t) \right].$$

A first-order homogeneous (or stationary) HMM uses a transition matrix $\mathbf{\Gamma}$ to characterize $P(S_t|S_{t-1})$ where $\mathbf{\Gamma}$ has entries $\gamma_{ij} = P(S_t = j|S_{t-1} = i)$. The homogeneity can be relaxed by allowing the transitions to depend on additional observed variables. Let $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$ be a sequence of observed input vectors, one for each data vector. One way to extend an HMM to

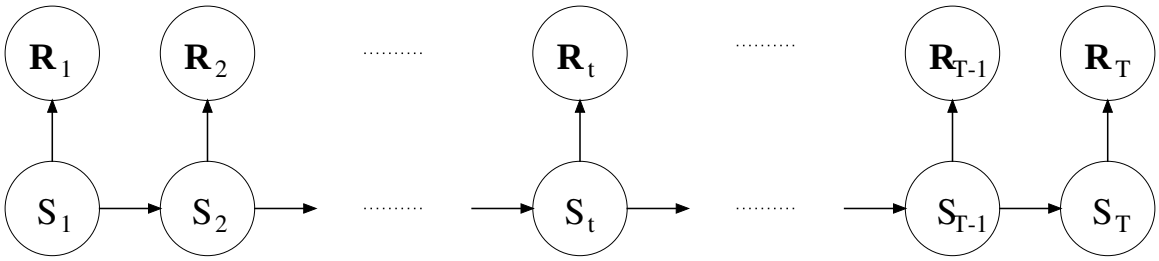

Figure 1: Graphical model interpretation of a hidden Markov model

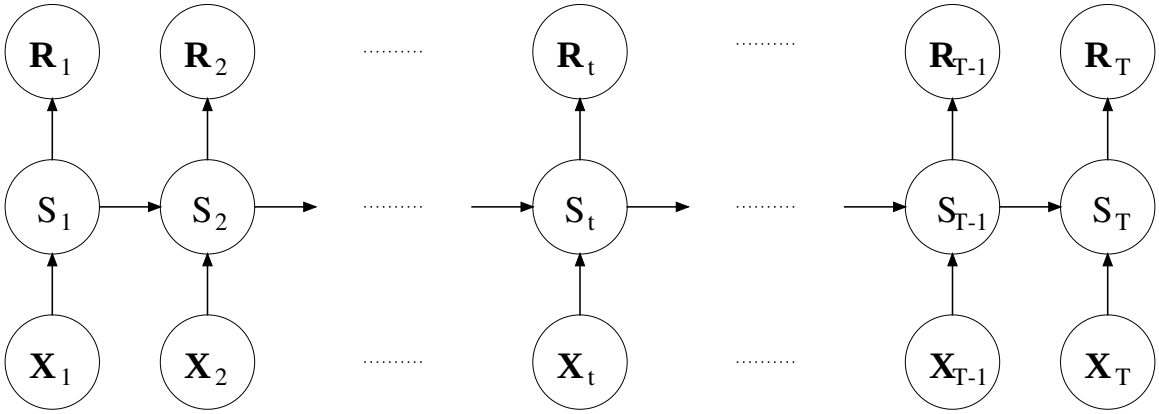

Figure 2: Graphical model interpretation of a nonhomogeneous hidden Markov model

nonhomogeneous HMM (NHMM) [1] is to make S_t dependent on both S_{t-1} and $\mathbf{X}_t$:

$$l = \log P(\mathbf{R}|\mathbf{X}) = \log \sum_{\mathbf{S}} \left[P(S_1|\mathbf{X}_1) \prod_{t=2}^{T} P(S_t|S_{t-1}, \mathbf{X}_t) \right] \left[\prod_{t=1}^{T} P(\mathbf{R}_t|S_t) \right].$$

A graphical model interpretation of NHMM is provided in Figure 2.

3 Package Details and Installation

The package has been developed in C++ on the RedHat 9.0/Fedora 3 Linux platform. The initial development has been performed on the Sun Solarix 5.8 platform. The code has been tested on both platforms, and no crashes have been experienced on either platform, provided the parameter file is specified properly. The authors cannot guarantee compatibility for other platforms, although we have used it under MacOS X without difficulty (with make -fmakefile.linux).

The package consists of the source code, licence file, text README file with the current set of options, development update notes, a number of perl scripts, and a directory with example parameter files and data sets. The package contains one makefile for each Linux and Solaris platform. The source code is compatible with the g++ compiler on Linux (without warnings) and CC compiler for Sun (with a few minor warnings). Depending on the configuration of your system, you may have to tweak the makefile to have the program compile. We also recommend adding the directory with the binaries and the scripts (bin) to the path in the environment file.

Some of the algorithms implemented in the toolbox have been described in [2], [4], and [3]. More details will be provided in a future release of an expanded manual.

4 Running the program

In order to run the software, the user must first specify the options or *parameters* with which the program will run. One of the flipsides of the flexibility of the software is that the number of tunable parameters is quite large and cannot be easily passed in a command line. Instead, the program expects to be passed a parameter file with the values of parameters. The command line to run the program is

`mvnhmm` $\langle parameterfile \rangle$ [*seed*]

Seed specifies the initialization seed for the random number generator and is set to **0** if omitted. The ability to specify the random number generator seed is crucial for the ability to reproduce the results.

There are four types of parameters passed to the program in the parameter file (options covered in this manual are typed in **boldface**):

1. Mode to run the program in: **action**, **xval_type**, **examples_out**.
2. Specifics and auxiliaries for each mode: **num_restarts**, **num_simulations**, **analysis**, em_precision, em_verbose, initialization, filling.
3. Model related parameters: **num_states**, **model_type**, **emission**, **input_distribution**
4. Data file related parameters: **data**, **num_discrete_data_components**, **num_real_data_components**, **num_data_sequences**, **data_sequence_length**, **input_filename**, num_discrete_input_components, **num_real_input_components**, **num_input_sequences**, **input_sequence_length**, **model_filename**, num_models, **output**, state_filename.

In this document, we will only describe the most essential options while leaving more obscure ones to the full manual. The specification file allows comments beginning with # and ending at the end of the line. Empty lines are ignored.

4.1 Commonly Used Options

action ⟨**type**⟩ — specifies what task the program should perform. The values for the type of action are:

- **learn** – learns the values of the parameters for the model given the data (default for option **action**);
- **viterbi** – finds the most likely sequence of hidden states for each data sequence given the model;
- **ll** – calculates the log-likelihood of the data given the model;
- **simulation** – simulates the data given the model;
- **analysis** – calculates a summary statistic of the data (does not require a model specification).

model_type ⟨**type**⟩ — type of model to use. Possible model types are:

- **hmm** — homogeneous hidden Markov model (default for option **model_type**);
- **nhmm** — non-homogeneous hidden Markov model with transitions dependent on the values of the input variables (or covariates). NHMM also requires the specification of the input distribution (**input_distribution**) and the specification of the file with input variables (**input_filename**).

All model types require the specification of the type of emission probability distribution (distribution of observations given hidden states) using **emission**.

num_states ⟨**positive integer**⟩ — number of hidden states. The default value is **2**. While models with 1 hidden state value are allowed, the resulting models are not truly HMMs. As a word of caution, models with large number of values will take a long time to train.

emission ⟨**distribution type**⟩ — specification of the probability distribution type for each hidden state (assumed the same for all states). It is possible to specify a large number of distributions including nested distributions and mixtures. Mentioned below are just a few examples. Assume $\mathbf{R}_t = \boldsymbol{r}_t = \left(r_t^1, \ldots, r_t^M\right)$.

- **independent** ⟨**num$_1$**⟩ **bernoulli** ⟨**num$_2$**⟩ — conditionally independent Bernoulli (coin-toss) distribution with num_2 outcomes (0 through $num_2 - 1$) on num_1 variables ($num_1 = M$),

$$P\left(\boldsymbol{r}_t | S_t = i\right) = \prod_{m=1}^{M} P\left(r_t^m | S_t = i\right) = \prod_{m=1}^{M} p_{imr_t^m};$$

- **chow-liu** ⟨**num$_1$**⟩ ⟨**num$_2$**⟩ – tree-structured distribution with num_2 outcomes (0 through $num_2 - 1$) of num_1 variables (see [3] or [2] for more details);
- **gaussian** ⟨**num**⟩ — normal distribution with full covariance on num data variables,

$$P\left(\boldsymbol{r}_t | S_t = i\right) = (2\pi)^{-\frac{M}{2}} \left|\boldsymbol{\Sigma}_i\right|^{-1} \exp\left(-\frac{1}{2}\left(\boldsymbol{r}_t - \boldsymbol{\mu}_i\right)\boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{r}_t - \boldsymbol{\mu}_i\right)^t\right);$$

- **independent** ⟨**num$_1$**⟩ **chain-bernoulli** ⟨**num$_2$**⟩ — conditionally independent conditional on the previous observed value Bernoulli (coin-toss) distributions with num_2 outcomes (0 through $num_2 - 1$) on num_1 data variables,

$$P\left(\boldsymbol{r}_t | S_t = i, \mathbf{R}_{t-1} = \boldsymbol{r}\right) = \prod_{m=1}^{M} P\left(r_t^m | S_t = i, \boldsymbol{r}^m\right) = \prod_{m=1}^{M} p_{imr^m r_t^m};$$

- **conditional-chow-liu** ⟨**num$_1$**⟩ ⟨**num$_2$**⟩ – tree-structured conditional distribution with num_2 outcomes (0 through $num_2 - 1$) of num_1 variables (see [3] or [2] for more details);
- **independent** ⟨**num$_1$**⟩ **delta-exponential** ⟨**num$_2$**⟩ — conditionally independent num_2-component mixtures of Dirac delta function at 0 and $num_2 - 1$ exponential function on num_1 data variables,

$$\begin{aligned} P\left(\boldsymbol{r}_t | S_t = i\right) &= \prod_{m=1}^{M} P\left(r_t^m | S_t = i\right) = \prod_{m=1}^{M} a_{im} \text{ where} \\ a_{im} &= \begin{cases} p_{im0} & r_t^m = 0, \\ \sum_{c=1}^{num_2 - 1} p_{im1} \lambda_{imc} \exp\left(-\lambda_{imc} r_t^m\right) & r_t^m > 0. \end{cases} \end{aligned}$$

- **independent** ⟨**num$_1$**⟩ **delta-gamma** ⟨**num$_2$**⟩ — conditionally independent num_2-component mixtures of Dirac delta function at 0 and $num_2 - 1$ Gamma function on num_1 data variables,

$$\begin{aligned} P(\boldsymbol{r}_t|S_t = i) &= \prod_{m=1}^{M} P(r_t^m|S_t = i) = \prod_{m=1}^{M} a_{im} \text{ where} \\ a_{im} &= \begin{cases} p_{im0} & r_t^m = 0, \\ \sum_{c=1}^{num_2-1} p_{im1} \frac{\beta_{imc}^{\alpha_{imc}} \left(r_t^m\right)^{\alpha_{imc}-1} \exp\left(-\beta_{imc} r_t^m\right)}{\Gamma(\alpha_{imc})} & r_t^m > 0. \end{cases} \end{aligned}$$

input_distribution ⟨**distribution type**⟩ – specification of the probability distribution influencing the hidden state transition matrix given the input variables. Currently, only a few of input distributions are considered and implemented:

- **transition-logistic** ⟨**num_states**⟩ ⟨**num_input_variables**⟩ — a variation of polytomous (multinomial) logistic distribution used with nonhomogeneous hidden Markov models to describe a nonhmogeneous transition matrix. *num_states* is the number of hidden states, and *num_input_variables* is the number of input variables,

$$P(S_t = j|S_{t-1} = i, \mathbf{X}_t = \mathbf{x}) = \frac{\exp\left(\sigma_{ij} + \boldsymbol{\rho}_j \mathbf{x}^t\right)}{\sum_{k=1}^{K} \exp\left(\sigma_{ik} + \boldsymbol{\rho}_k \mathbf{x}^t\right)}$$

  where $K = num\_states$.

data ⟨**filename**⟩ — file containing the data set ot be analyzed. The data file is assumed to be a space or tab delimited file of vectors (see Section 4.2 for more details). Must be accompanied by **num_discrete_data_components**, **num_data_sequences**, and **data_sequence_length**.

num_discrete_data_components ⟨**non-negative integer**⟩ — number of discrete-valued components in each data vector (default is **0**).

num_data_sequences ⟨**positive integer**⟩ — number of data vector sequences. All sequences are assumed to be of the same length.

data_sequence_length ⟨**positive integer**⟩ — length of each data vector sequence.

input_filename ⟨**filename**⟩ — file location of the input variables. The input file is assumed to be of the same format as the data file, space or input delimited file of vectors. Must be accompanied by **num_real_input_components**, **num_input_sequences**, and **input_sequence_length**.

num_real_input_components ⟨**non-negative integer**⟩ — number of real-valued components in each input vector (default is **0**).

num_input_sequences ⟨**positive integer**⟩ — number of input vector sequences. All sequences are assumed to be of the same length. The number of input sequences must match the number of data sequences.

input_sequence_length ⟨**positive integer**⟩ — length of each input vector sequence. While redundant, the length of input sequencces must match the length of data sequences.

output ⟨**filename**⟩ — the location of file where the results of the program run will be written. Note that the type of output file will depend on the **action**. If no filename is provided, the results are outputted to the screen.

model_filename ⟨**filename**⟩ — the location of the file with the model parameters. Required for action types **viterbi**, **ll**, and **simulation**.

num_restarts ⟨**positive integer**⟩ — number of random restarts used during the parameter estimation. This option is used with action type **learn**. Default valued is **1**.

num_simulations ⟨**positive integer**⟩ — number of times a data set is simulated from a model. The number of sequences in each set is specified by **num_data_sequences**, and the length of each simulated sequences is specified by **data_sequences_length**. Default value is **1**.

xval_type ⟨**type**⟩ – cross-validation mode used with the software. There are two options:

- **none** – no cross-validation used (default option). All of the data sequences are used in all runs.
- **leave_n_out** – leave-n-out cross-validation. Assuming N data sequences, each run consists of N/n runs with k-th run ($k = 1, \ldots, N/n$) using sequences $1, \ldots, (k-1) \times n, k \times n + 1, \ldots, N$. This option requires the specification of the number of sequences to leave out via the parameter **examples_out**.

examples_out ⟨**positive integer**⟩ – number of sequences to leave out with cross-validation type **leave_n_out**. The number of left out sequences must be between 1 (default) and a half of the total number of sequences.

analysis ⟨**type**⟩ — the type of data statistics to collect for action **analysis**. Currently, can only collect statistics for binary data sets. Possible type values are:

- **mean** — proportion of **1**'s for each variable;
- **correlation** — Pearson's correlation between each pair of variables;
- **persistence** — proportion of **0**s after a **0** and **1**s after a **1** for the same variable as computed for each variable;
- **dry** – histogram of consecutive occurrences of **0**s for the same variable (dry spell runs) for each variable;
- **wet** – histograms of consecutive occurrences of **1**s for the same variable (wet spell runs) for each variable;
- **information** – mutual information values for each pair of variables;
- **logodds** – log-odds ratio for each pair of variables.

4.2 File formats

There are a few different file types used with the toolbox. Although types havve some significant differences, the same general rules used when the files are read in by the program. All files are stored and recorded in ASCII format. The files may contain comments as the toolbox ignores all text in the files after the character "#". Except for the specification file, only numeric values can be processed. Spaces, tabs, and carriage return all serve as separators between numeric values. The toolbox will ignore all extra separators. Below, we provided some additional information about most frequent file types.

- **Data** file is assumed to consist of sequences of vectors. This type of file is passed to the program using options **data** or **input_filename**, and is the type of file generated in the **simulation** mode. Each vector is in turn assumed to consist of multiple components. When the program reads in the data file, it reads in all components of the first vector, then all components of the second vector, and so on until the whole sequence of vectors is read. Then the subsequent sequence of vectors is read in, and so on until all sequences are read in. The number of sequences, lengths of sequences, and the number of components are specified in the parameter file. Values for categorical variables are assumed to be represented by natural numbers (e.g., a binary variable would take on values 0 and 1). The toolbox will treat NaN entries as missing.
- **Model** file contains the values of the model's parameters. This is the type of file generated by the program in the **learn** mode (see example in Section 5), and the type of file used in other modes and passed to the program using **model_filename** option. The file does **not** contain the specification of the model, but only the parameter values. It will contain comments indicating what parameter the value corresponds to, the index information allowing the program to match the parameter value to the corresponding vector component in the data, and the values of the parameters. The bottom four lines of the file provide some basic statistics of the learning: log-likelihood (and scaled log-likelihood) of the data given the model, Bayes Information Criterion, and the number of failed runs. The run is flagged as failed if the log-likelihood of the model is infinite or NaN. Most likely reasons for failure are numerical instabilities and overfitting.
- **Analysis** file contains statistics of the data. It is generated in the **analysis** mode.
- **State sequence** file contains most likely state sequences for the data given the model. It is generated in the **viterbi** mode and consists of space-delimited sequences (rows) of state indicators (integers from 0 to the number of hidden states minus one), one for each vector sequence of the data set.

5 Examples

Directory `examples` contains directories with several data sets, parameter files, and an example of a script run.

The subdirectory `param_files` contains several example parameter files for various set-ups: model learning (i.e. training), most likely sequence estimation, simulation, computation of data log-likelihood, and data analysis. The filenames given to the parameter files indicate the mode in which the program is run, the model type, and the emission type. File `template` contains the template with all currently available options for the software.

The subdirectory `outputs` contains outputs of the program runs with some of the parameter files with default seed.

The subdirectory `models` contains models used in some of the parameter files.

The subdirectory `data` contains some daily rainfall data sets from the state of Ceará, Northeast Brazil (see [4] for some additional info). It also contains data simulated from an HMM with Gaussian emissions (model `hmm_gaussian`).

5.1 Model Learning Example

Below is an example of using the toolbox to learn the parameters of the model. This example can be found in `examples/param_files/learn_hmm_independent`. The model is an HMM with 4 hidden states with emission probabilities modeled as 10 conditionally independent binary (0 or 1) Bernoulli distributions. The data set is located in directory `examples/data/brazil_occurrence_data` and consists of 24×90 sequences of 10 binary numbers. The algorithm uses 10 different starting points and outputs the result (model parameters) of the best of 10 runs into the file `examples/outputs/learn_hmm_independent`. To run the toolbox with this example, change the directory to `examples/param_files/` and run `mvnhmm learn_hmm_independent`.

```
# Action
action learn

# Type of the model
model_type hmm

# Number of hidden states
num_states 4

# Emission distribution
emission
independent 10 bernoulli 2

# Data file
data ../data/brazil_occurrence_data

# Number of finite-valued vector components for the data
num_discrete_data_components 10

# Number of output data sequences
num_data_sequences 24

# Length of each sequence
data_sequence_length 90

# Output file
output ../outputs/learn_hmm_independent
```

```
# Number of random restarts
num_restarts 10
```

5.2 Example of Model Parameters File

The excerpts from the result of the run of the parameter file (with default seed) from the previous subsection is provided below. For HMMs, the parameter file contains $P(S_1)$ (first line of parameters), $P(S_t|S_{t-1})$ (the transition matrix), and then $P(\mathbf{R}_t|S_t = i)$, the parameters for each hidden state i (under `State #`i). Each state consists Bernoulli distributions for each of 10 vector components (0 through 9) with two probabilities ($P(R_{tm}|S_t) = 0$ and 1) for each component. Interpretation of the model parameters depends on application. Here (precipitation modeling), probabilities of 1 (second column in the probabilities in each of hidden states) for a component corresponds to probability of rain for the respective rain station given the hidden state. For example, the probability of rain in state 1 for station 6 is 7.370570344412e-02 ≈ 0.0737.

```
# This file contains the parameters of the HMM
# First state probabilities:
2.303191708043e-03 6.667565498714e-01 1.105932880693e-01 2.203469703513e-01

# Transition matrix:
5.659160742980e-01 1.426541325258e-01 1.227510352329e-01 1.686787579433e-01
1.456102706329e-01 6.795820646524e-01 1.506500167471e-01 2.415764796763e-02
1.819946284549e-01 1.910636564195e-01 4.242538459204e-01 2.026878692051e-01
1.648174395276e-01 1.202733861656e-02 1.196452640076e-01 7.035099578483e-01

# Parameters for the emission distributions for different possible hidden states:
# State #1:
0
7.754067036457e-01 2.245932963543e-01
1
4.539662253961e-01 5.460337746039e-01
2
8.167467708670e-01 1.832532291330e-01
3
7.310120912977e-01 2.689879087023e-01
4
3.760346199131e-01 6.239653800869e-01
5
9.262942965559e-01 7.370570344412e-02
6
5.663832499786e-01 4.336167500214e-01
7
6.868134023556e-01 3.131865976444e-01
8
3.147058695387e-01 6.852941304613e-01
9
3.041020582004e-01 6.958979417996e-01

# State #2:
```

```
0
9.645380144018e-01 3.546198559824e-02
1
9.207729680219e-01 7.922703197806e-02
2
8.988931224789e-01 1.011068775211e-01
3
9.708213567685e-01 2.917864323147e-02
...
...
...
# Log-likelihood of the data set: -1.215351861061e+04
# Per-dimension log-likelihood of the data set: -5.626628986392e-01
# Bayes Information Criterion score: 2.472931971375e+04
# Failure rate: 0 out of 10.
```

5.3 Data Simulation Example

Below is an example of using the toolbox to simulate data from a given model. (This parameter file can be found in `examples/param_files/sim_hmm_independent`.) The model is an HMM with 4 hidden states with emission probabilities modeled as 10 conditionally independent binary (0 or 1) Bernoulli distributions. The parameters of the model can be found in `examples/models/hmm_independent`. The program will simulate 24×2 sequences of 10-component binary vectors of length 90 each into the file `examples/outputs/sim_hmm_independent`. To run the toolbox with this example, change the directory to `examples/param_files/` and run `mvnhmm sim_hmm_independent`.

```
# Action
action simulation

# Type of the model
model_type hmm

# Number of hidden states
num_states 4

# Emission distribution
emission
independent 10 bernoulli 2

# Number of finite-valued vector components for the data
num_discrete_data_components 10

# Number of output data sequences
num_data_sequences 24

# Length of each sequence
data_sequence_length 90
```

```
# Output file
output ../outputs/sim_hmm_independent

# Model file name
model_filename ../models/hmm_independent

# Number of simulations per output sequence
num_simulations 2
```

5.4 Model Evaluation Scripts

The subdirectory `batch` contains a sample run of one of the batch perl scripts, `mvnhmm-batch-simple.perl`. These scripts allow several actions of the program to be consecutively, over a range of numbers of hidden states. (**Note:** No knowledge of perl needed to run the scripts, only to modify the scripts.) To run the script, first create subdirectories `data` and `outputs`. Place the data files used for the modeling in `data`. To view the parameters used with scripts, enter the script name without inputs. The results in the `outputs` subdirectory are a result of a run `mvnhmm-batch-simple.perl hmm 10 independent 24 90 0 0 5e-05 50 500 2 6`. This batch script trains HMM models on the binary data with 24 sequences of 90 10-dimensional observations each. Models with number of hidden states between 2 and 6 are considered. Each model is trained with 50 random restarts. For each model, 500 times the original data is simulated, and the statistics of that data are calculated.

6 Other Features

Description of other features can be found in the full manual (to appear soon). File `README` is the most up-to-date document on the available features and describes full functionality of the code in a compact but dense manner. `examples/param_files` contains a number of examples covering most of the code's functionality. Readers are encouraged to try them out.

7 Acknowledgements

This work was supported by the Department of Energy under grant DE-FG02-02ER63413. The daily rainfall data were kindly provided by the Brazilian State of Ceará's Foundation for Meteorology and Water Resources (FUNCEME; `http://www.funceme.br`). The author thanks P. Smyth of University of California, Irvine and A.W. Robertson of International Research Institute (IRI) for Climate Prediction for their invaluable input in the development of the software.

References

[1] J. P. Hughes, P. Guttorp, and S. P. Charles. A non-homogeneous hidden Markov model for precipitation occurrence. *Journal of the Royal Statistical Society Series C Applied Statistics*, 48(1):15–30, 1999.

[2] S. Kirshner. *Modeling of Multivariate Time Series Using Hidden Markov Models*. PhD thesis, University of California, Irvine, March 2005.

[3] S. Kirshner, P. Smyth, and A. W. Robertson. Conditional Chow-Liu tree structures for modeling discrete-valued vector time series. In M. Chickering and J. Halpern, editors, *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 317–324. AUAI Press, 2004.

[4] A. W. Robertson, S. Kirshner, and P. Smyth. Downscaling of daily rainfall occurrence over Northeast Brazil using a hidden Markov model. *Journal of Climate*, 17(22):4407–4424, November 2004.