Learning mixture models – courseware for finite mixture models of multivariate Bernoulli distributions

Jaakko Hollmén Jaakko.Hollmen@tkk.fi
Tapani Raiko Tapani.Raiko@tkk.fi
Department of Information and Computer Science, Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
`http://www.cis.hut.fi/jhollmen/BernoulliMix`

Abstract

Teaching of machine learning should aim at the readiness to understand and implement modern machine learning algorithms. Towards this goal, we often have course exercises involving the student to solve a practical machine learning problem involving a real-life data set. The students implement the programs of machine learning methods themselves and gain deep insight on the implementation details of the method. The downside of this approach is that time is devoted on implementation aspects rather than machine learning. Complementary to this approach, we have designed a machine learning course exercise on a ready implementation of the Expectation-Maximization (EM) algorithm for finite mixture distributions of multivariate Bernoulli distributions. We describe BernoulliMix — a program package with a set of teaching examples and exercises and report on the preliminary experiences in our class of machine learning students. The BernoulliMix package will be available under a liberal open source license.

1. Introduction

On machine learning courses, theoretical background is taught in classroom sessions with the help of an lecturer, and pen-and-paper exercises are completed and checked with the course assistant during the exercise sessions. In order to gain experience in solving practical machine learning problems, courses have term projects, where students practice solving a machine learning problem of practical importance. The term project usually involves a task and a data set, and the students will implement a solution, run the experiments, and write a report resembling a conference article. On our courses with 5 ECTS credit units, the practical term project is estimated to take 1 ECTS worth of work. The course is taught for the advanced Master's level students and beginning post-graduate students and is part of the Macadamia Master's program in data mining and machine learning (Raiko et al., 2008). Traditionally, the students have the task of implementing a solution to the machine learning problem from scratch. The most popular tool for the implementation has been Matlab (Mathworks, 1994). Although Matlab supports quick prototyping, a large part of the implementation work goes into handling inputs and outputs and other practical annoyances and may divert the focus of the student from machine learning to "just getting the programs to work" or particular programming details on the chosen environment.

In enable the students to focus most of their effort on machine learning and modeling, rather than programming details, we have tried an alternative approach. We provide a ready implementation that has been tested and well documented. This allows students to concentrate on the machine learning aspects. The term project contains problems, from which some can be solved just by running the software, and some require extending the software in some way, say implementing cross-validation. We believe that the program package may appeal to students in both machine learning (Bishop, 2006) and data mining (Hand et al., 2001).

In Section 2, we describe the history and the design decisions behind BernoulliMix software package, the individual programs implemented and continue by explaining the design and structure of documentation in Section 3. We discuss the BernoulliMix package in Section 4 in light of other possible solutions. In Section 5, we summarize our preliminary experiences about BernoulliMix.

2. BernoulliMix package

The BernoulliMix program package started out as bits and pieces of an implementation while working on earlier research contributions (Hollmén et al., 2003; Tikka et al., 2007). Recently, additional programs were implemented and the whole converged to a package with documentation and most importantly, examples and exercises for the machine learning courses in mind. According to (Brooks, 1995), the full working software system with documentation may take as much as ten times the effort of the simple straightforward implementation of the core functionality. In the current effort, we have experienced the same, if not greater overhead on top of the simple implementation effort.

Important aspects in the design of BernoulliMix courseware package has been the compatibility with a wide variety of different computing platforms, correctness and efficiency aspects, which is especially important for large-scale data mining applications, but maybe most importantly, the simplicity of use for the students. Simplicity is also supported by the choice of focusing on a rather narrow subset of material on a typical machine learning course, namely the finite mixture models. Therefore, each of the typical tasks in a machine learning setting has been implemented as a separate program, each accessible as a command line program executed in the Linux command shell. These tasks include the initialization of the parameters (Section 2.1), likelihood calculation with given data (Section 2.2), learning from data (Section 2.3), sampling from the mixture model (Section 2.4) and clustering data with the mixture model (Section 2.5).

The mixture model used in the BernoulliMix package is a finite mixture model of multivariate Bernoulli distributions (Wolfe, 1970), solely concentrating on modeling of 0-1 (zero-one) data. The likelihood of the observed data is readily calculated with

$$P(\mathbf{x}) = \sum_{j=1}^{J} \pi_j \, P(\mathbf{x} \mid \boldsymbol{\theta}_j) = \sum_{j=1}^{J} \pi_j \prod_{i=1}^{d} \theta_{ji}^{x_i} (1-\theta_{ji})^{1-x_i},$$

where the data vector is $\mathbf{x} = (x_1, \ldots, x_d)$, the data is binary $x_i \in \{0,1\}$ and the mixture model is parameterized by the mixing coefficients $\pi_j, j = 1, \ldots, J$ and the parameters for the component distributions $\boldsymbol{\theta}_j = (\theta_{j1}, \theta_{j2}, \ldots, \theta_{jd})$, $\theta_{ij} \in [0,1]$

2.1. Initialization of the mixture model

The parameters of the model are initialized by sampling from the uniform distribution of desired range. The mixing coefficients are initialized to be equal. The model is written to a file or standard output. An example command is

```
./bmix_init --clusters 2 --data-dim 3 \
     --model tiny.model
```

which initializes a model with 2 component distributions for modeling 3-dimensional 0-1 data and writes the model to a file.

2.2. Likelihood calculation

The likelihood of a data with a given mixture model can be calculated either for the whole data or for each data vector separately. Following the previous example, an example command

```
./bmix_like --data my.data --model tiny.model
```

calculates the likelihood of the data in the data file my.data with the previously initialized model.

2.3. Learning from data with EM algorithm

The learning of parameters in the framework of maximum likelihood can be performed with the Expectation-Maximization (EM) algorithm (Wolfe, 1970; Dempster et al., 1977). The following command trains (or learns) a model with the EM algorithm.

```
./bmix_train --data my.data --model-in \
     tiny.model --model trained.model \
     --iterations 50
```

2.4. Sampling data from the mixture model

As the model is generative in nature, it is natural to include a program to sample data from the mixture model. The command

```
./bmix_sample --model trained.model \
     --number-of-samples 100
```

samples 100 data samples from the model trained.model with the ancestral sampling scheme with the results printed on standard output.

2.5. Clustering data with the mixture model

Mixture model can be used in clustering by associating a component distribution with a cluster. Clusters are defined according to a maximum a posteriori rule. The command

```
./bmix_cluster --data my.data --model \
     trained.model --cluster 1
```

produces an output of data belonging to cluster 1.

3. Documentation: usage, examples and exercises

The documentation of the BernoulliMix program package has been written in Texinfo format, to make it possible to produce both on-line documentation and a printed manuals in dvi, ps and pdf formats easily. In the documentation, there is very little background theory on the models, rather references to the literature and available course books are given. It is stressed that the documentation is not designed to be any kind of replacement for a good text book on the topic. This underlines the role of the package as an auxiliary material to existing teaching material, not as a replacement to it.

In writing the documentation, we have concentrated on three aspects: the description of the usage of the programs, usage examples with functional descriptions, and the exercises. Each program is presented separately with the technical descriptions of command line options. Both long and short formats are available. The presentation of usage is followed by a handful of examples with the task described in the paragraph. This presentation ties together the actual execution of the programs and the machine learning modeling world taught in the classrooms. The examples are followed by exercises that form the core of the term project. The term project is to complete all the exercises. The easiest exercises follow closely the presented examples, but often the students are required to explain their findings more thoroughly. The most tedious one of the exercises is a model selection problem in modeling DNA copy number amplifications with a mixture model (Tikka et al., 2007; Myllykangas et al., 2008), where the student is asked to select an appropriate number of component distributions in a cross-validation setting by performing a 5-fold-cross-validation repeated 10 times for models ranging from 2 component distributions to 30.

4. Discussion

The counterargument raised by one the reviewers of this contribution was that it might be preferable to use one of the existing platforms for a machine learning course exercise rather than introduce yet another system and implement everything from scratch. The design goal of the BernoulliMix is to enable the student to concentrate on modeling and machine learning aspects in the term project, and therefore we should minimize the effort to learn a new environment or programming language. Matlab is widely known due to its adoption on the earlier mathematics courses as the tool. Other recognized programming tools among the students are quite varied, including various programming languages and mathematical programming environments. One environment that every student has been taught to use is the shell environment of the UNIX/Linux type of operating systems. This has guided us to introduce a command-line interface executed in a shell environment and the implementation to be such that it is easily compiled on a wide number of computing environments with minimal effort. Also, we tend to lean towards an open source implementation, which makes it possible to investigate and extend the program code, including benefits claimed by others (Sonnenburg et al., 2007). The current implementation is written in C programming language (Kernighan & Ritchie, 1988). This provides compatibility with a wide number of platforms and an efficient implementation, especially needed in the context of data mining. On top of the existing programs, it should be relatively easy to write interface functions for systems like Matlab (Mathworks, 1994) and R (R Project, 1997). In order to be realistically extensible by all students, the documentation should also include a description of the application program interface (API).

5. Summary and Conclusions

We have described our approach to a term project on a machine learning course: instead of making a full implementation of the programs needed to solve a machine learning problem, the students work with the ready implementation and concentrate on the modeling and experimentation aspect. We have described a courseware package and our preliminary experiences with our current class of machine learning students, who are completing the exercise. Our experience is that the students have been helpful in providing technical feedback on compiling the programs on different operating systems, and on the technical implementations of the programs. Most active students have suggested improvements as regards to the general cross-platform build commands that make the program package available in many different platforms with the C compiler.

Others have argued that there is a need for open source software in machine learning (Sonnenburg et al., 2007). When using open source packages in teaching, the students will get used to using and possibly extending existing software rather than rewriting everything from scratch. It also promotes the spirit of sharing software in open source form.

After the exercise has been completed in the end of this semester, feedback will be put in refining the exercises and the implementation. Also, time estimates for completing the exercises may be more accurately given. After the feedback round, the implementation and the documentation will be made available to the machine learning and data mining community with a liberal open source license.

Acknowledgments

We thank the anonymous reviewers for the fruitful comments on the original submission. We are grateful to Paul Grouchy and Niko Vuokko for extensive feedback on the package.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information Science and Statistics. Springer, New York.

Brooks, F. (1995). *The mythical man-month: Essays on software engineering*. Addison-Wesley. 20th anniversary edition edition.

Dempster, A. P., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. Adaptive Computation and Machine Learning Series. MIT Press.

Hollmén, J., Seppänen, J. K., & Mannila, H. (2003). Mixture models and frequent sets: combining global and local methods for 0-1 data. *Proceedings of the Third SIAM International Conference on Data Mining* (pp. 289–293).

Kernighan, B. W., & Ritchie, D. M. (1988). *The C programming language*. Prentice Hall. Second edition edition.

Mathworks (1994). Matlab — the language of technical computing. `http://www.mathworks.com/products/matlab/`.

Myllykangas, S., Tikka, J., Böhling, T., Knuutila, S., & Hollmén, J. (2008). Classification of human cancers based on dna copy number amplification patterns. *BMC Medical Genomics*. in press.

R Project (1997). The R project for statistical computing. `http://www.r-project.org/`.

Raiko, T., Puolamäki, K., Karhunen, J., Hollmén, J., Honkela, A., Kaski, S., Mannila, H., Oja, E., & Simula, O. (2008). Macadamia: Master's programme in machine learningi and data mining. *Teaching Machine Learning: Workshop on open problems and new directions*. Saint-Étienne, France.

Sonnenburg, S., Braun, M. L., Ong, C. S., Bengio, S., Bottou, L., Holmes, G., LeCun, Y., Müller, K.-R., Pereira, F., Rasmussen, C. E., Rätsch, G., Schölkopf, B., Smola, A., Vincent, P., Weston, J., & Williamson, R. (2007). The need for open source software in machine learning. *Journal of Machine Learning Research*, *8*, 2443–2466.

Tikka, J., Hollmén, J., & Myllykangas, S. (2007). Mixture modeling of DNA copy number amplification patterns in cancer. *Proceedings of the 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)* (pp. 972–979). San Sebastián, Spain: Springer-Verlag.

Wolfe, J. W. (1970). Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, *5*, 329–350.