

Parameter Reusing in Learning Latent Class Models

Gytis Karčiauskas, Finn V. Jensen

Aalborg University
{*gytis,fvj*}@*cs.auc.dk*

Tomáš Kočka¹

Prague University of Economics
kocka@lisp.vse.cz

1 Introduction

Latent class analysis (LCA) [13], [12] is a method for finding classes of similar cases from multivariate categorical data. The data is assumed to be generated by a *latent class* (LC) model. An LC model consists of a hidden *class variable* and observed *manifest variables*. Each state of the class variable corresponds to a different component (class). Manifest variables are assumed to be conditionally independent given the class variable. The *model parameters* consist of the prior probability distribution for the class variable and the conditional probability distributions for each manifest variable given the class variable. Usually the model parameters are learned by using the EM algorithm [5]. The well-known problem of the EM algorithm is that local rather than global maximum solutions can be found [12], [18]. A simple and standard way of dealing with this problem is to run the EM algorithm from many different starting points [4]. However, often a high number of starting points is needed to find the global maximum solution, thus making the algorithm computationally very expensive.

One could try to get the global maximum solution in much shorter time by using information about the parameters of the LC models with fewer components. The idea of reusing the parameters has already been applied in learning various finite mixture models [21], [20], [15], [19], [9], [11], [6]. There have also been proposed other approaches for escaping local maximum in EM [7], [8] or making EM faster [16], [2], [17].

In this paper, we propose an algorithm for learning LC models by repeatedly splitting a component. Verbeek et al. [20] use the same idea for continuous data, where Principal Component Analysis is used for initialising the new components. In our case, the data is categorical, and we split a component randomly. We provide a theoretical justification of our approach. In the experiments on various real-world data our algorithm performs better than the standard one.

¹Supported by GA CR grant n. 201/02/1269.

2 Parameter Reusing

2.1 Notation and Definitions

A variable is denoted by an upper-case letter (for example, D_i), a state of a variable by a lower-case letter (for example, d_i). A vector of states for a set of variables is denoted by a bold upper-case letter (for example, \mathbf{d}). Training data over discrete variables D_1, \dots, D_k is $\mathbf{D} = (\langle \mathbf{d}_1, n_1 \rangle, \dots, \langle \mathbf{d}_N, n_N \rangle)$, where \mathbf{d}_i is a k -dimensional vector, n_i is an integer count, and $\mathbf{d}_i \neq \mathbf{d}_j$ for $i \neq j$. LC is a latent class model with components h_1, \dots, h_m . The probability of $\mathbf{d} = (d_1, \dots, d_k)$ given LC is $P_{LC}(\mathbf{d}) = \sum_{i=1}^m P_{LC}(h_i) P_{LC}(\mathbf{d}|h_i) = \sum_{i=1}^m P_{LC}(h_i) \prod_{j=1}^k P_{LC}(d_j|h_i)$, where P_{LC} is specified by the parameters of LC . The log-likelihood of data \mathbf{D} given model M is denoted as $LL(\mathbf{D}|M) = \sum_{i=1}^N n_i \ln P_M(\mathbf{d}_i)$.

$P(D_1, \dots, D_k|h_l)$ denotes the empirical joint probability distribution of D_1, \dots, D_k in the component h_l of LC . This distribution is computed from \mathbf{D} and LC :

$$P(D_1 = d_1, \dots, D_k = d_k|h_l) = \frac{n' P_{LC}(h_l|(d_1, \dots, d_k))}{\sum_{\langle \mathbf{d}, n \rangle \in \mathbf{D}} n P_{LC}(h_l|\mathbf{d})},$$

where n' is from $\langle (d_1, \dots, d_k), n' \rangle \in \mathbf{D}$. Similarly, for any $\mathcal{S} \subset \{D_1, \dots, D_k\}$, $P(\mathcal{S}|h_l)$ denotes the probability distribution obtained by marginalizing $\{D_1, \dots, D_k\} \setminus \mathcal{S}$ out of $P(D_1, \dots, D_k|h_l)$.

We say that model LC^* is obtained from model LC by *splitting* a component h_s if LC^* contains h_s and a new component that are both similar to h_s in LC , and all the other components are identical in both models. More formally, LC contains components h_1, \dots, h_m , LC^* contains components h_1, \dots, h_{m+1} and there exists $h_s \in \{h_1, \dots, h_m\}$ such that:

- $P_{LC^*}(h_i) = P_{LC}(h_i), P_{LC^*}(D_j|h_i) = P_{LC}(D_j|h_i), i = 1, \dots, m, i \neq s,$
- $P_{LC^*}(h_s) = P_{LC^*}(h_{m+1}) = \frac{1}{2}P_{LC}(h_s),$
- There exists small $\epsilon > 0$ such that $\|P_{LC^*}(D_j|h_s) - P_{LC}(D_j|h_s)\| < \epsilon, \|P_{LC^*}(D_j|h_{m+1}) - P_{LC}(D_j|h_s)\| < \epsilon, j = 1, \dots, k.$

2.2 Theoretical Properties of Component Splitting

Theorem 1 *If for given \mathbf{D} the log-likelihood $LL(\mathbf{D}|LC)$ is not maximal, then with probability 1 it is possible by splitting some component of LC to obtain a model LC^* such that $LL(\mathbf{D}|LC^*) > LL(\mathbf{D}|LC)$.*

Outline of proof. Since $LL(\mathbf{D}|LC)$ is not maximized, there exists a component h_l such that

$$P(D_1, \dots, D_k|h_l) \neq P_{LC}(D_1|h_l) \cdot \dots \cdot P_{LC}(D_k|h_l). \quad (1)$$

Then with probability 1 there exist variables D_a, D_b ($1 \leq a, b \leq k, a \neq b$) such that

$$P(D_a, D_b|h_l) \neq P_{LC}(D_a|h_l)P_{LC}(D_b|h_l). \quad (2)$$

Assume that $\forall i : P_{LC}(D_i|h_l) = P(D_i|h_l)$ (otherwise, the log-likelihood can be increased by simply taking parameters that make these equalities true). Then there exist states $\{a_1, a_2\}$ of D_a and states $\{b_1, b_2\}$ of D_b such that

$$\begin{aligned} & \frac{P(D_a = a_1, D_b = b_1|h_l)}{P_{LC}(D_a = a_1|h_l)P_{LC}(D_b = b_1|h_l)} + \frac{P(D_a = a_2, D_b = b_2|h_l)}{P_{LC}(D_a = a_2|h_l)P_{LC}(D_b = b_2|h_l)} \neq \\ & \frac{P(D_a = a_2, D_b = b_1|h_l)}{P_{LC}(D_a = a_2|h_l)P_{LC}(D_b = b_1|h_l)} + \frac{P(D_a = a_1, D_b = b_2|h_l)}{P_{LC}(D_a = a_1|h_l)P_{LC}(D_b = b_2|h_l)}. \quad (3) \end{aligned}$$

Let us produce model LC^* by splitting component h_l of LC into components h_l and h_{m+1} . Set the parameters of LC^* to be equal to the parameters of LC (and component h_{m+1} the same as component h_l) except:

$$P_{LC^*}(h_l) = P_{LC^*}(h_{m+1}) = \frac{1}{2}P_{LC}(h_l), \quad (4)$$

$$P_{LC^*}(D_v = v_1|h_l) = P_{LC}(D_v = v_1|h_l) + \sqrt{\epsilon}, \quad (5)$$

$$P_{LC^*}(D_v = v_2|h_l) = P_{LC}(D_v = v_2|h_l) - \sqrt{\epsilon}, \quad (6)$$

$$P_{LC^*}(D_v = v_1|h_{m+1}) = P_{LC}(D_v = v_1|h_l) - \sqrt{\epsilon}, \quad (7)$$

$$P_{LC^*}(D_v = v_2|h_{m+1}) = P_{LC}(D_v = v_2|h_l) + \sqrt{\epsilon}, \quad (8)$$

where $v \in \{a, b\}$ and $\epsilon \in \mathbb{R}$.

Let $\mathbf{D}' = \{\langle \mathbf{d}, n \rangle \in \mathbf{D} : D_a \in \{a_1, a_2\} \text{ and } D_b \in \{b_1, b_2\} \text{ in } \mathbf{d}\}$. If $\langle \mathbf{d}, n \rangle \notin \mathbf{D}'$, then $P_{LC^*}(\mathbf{d}) = P_{LC}(\mathbf{d})$. If $\langle \mathbf{d}, n \rangle \in \mathbf{D}'$, then $P_{LC^*}(\mathbf{d}) = P_{LC}(\mathbf{d}) + P_{LC}(h_l) \left(\prod_{\substack{j=1 \\ j \neq a, b}}^k P_{LC}(d_j|h_l) \right) s(d_a, d_b)\epsilon$, where $s(a_x, b_y) = \begin{cases} 1, & \text{if } x = y \\ -1, & \text{otherwise} \end{cases}$.

Then $LL(\mathbf{D}|LC^*) = LL(\mathbf{D}|LC)$ for $\epsilon = 0$, and

$$\begin{aligned} & \frac{\partial LL(\mathbf{D}|LC^*)}{\partial \epsilon}(0) = \left(\sum_{\langle \mathbf{d}, n \rangle \in \mathbf{D}} n P_{LC}(h_l|\mathbf{d}) \right) \cdot \\ & \left(\sum_{x \in \{1, 2\}, y \in \{1, 2\}} s(a_x, b_y) \frac{P(D_a = a_x, D_b = b_y|h_l)}{P_{LC}(D_a = a_x|h_l)P_{LC}(D_b = b_y|h_l)} \right). \end{aligned}$$

Because of Inequality 3, $\frac{\partial LL(\mathbf{D}|LC^*)}{\partial \epsilon}(0) \neq 0$. If $\frac{\partial LL(\mathbf{D}|LC^*)}{\partial \epsilon}(0) > 0$, we can make $LL(\mathbf{D}|LC^*) > LL(\mathbf{D}|LC)$ by taking ϵ small enough. Otherwise, if $\frac{\partial LL(\mathbf{D}|LC^*)}{\partial \epsilon}(0) < 0$, we can make $LL(\mathbf{D}|LC^*) > LL(\mathbf{D}|LC)$ by changing the sign before $\sqrt{\epsilon}$ in Equations 5-8 for $v = b$ and taking ϵ small enough. ■

So, if for \mathbf{D} large enough, the score² of an LC model is not maximal, then with probability 1 it is possible to increase the model score by splitting a component.

2.3 Algorithms for Learning LC Models

The algorithms described here take as an input multivariate categorical data that may contain missing values. The algorithms can run arbitrarily long, and the score of an LC model improves as the computation time increases.

Our algorithm, called *ParamReusing* and described in Figure 1, repeatedly tries to increase the number of components by splitting one of the current components. Even though it would be possible to split a component as described in Section 2.2, in practice it is more efficient to split it randomly. In step 2a, component h_s is split by making $P_{LC^*}(D_j|h_s) - P_{LC}(D_j|h_s) = \vec{r}_j$ and $P_{LC^*}(D_j|h_{m+1}) - P_{LC}(D_j|h_s) = -\vec{r}_j$. Each element of \vec{r}_j is a random number from $(-p; p)$, where $p \in \mathbb{R}$ is a small positive parameter. In step 2c, any penalized likelihood score can be used.

1. Let LC be a model with one component.
2. Repeat
 - (a) Produce set of models \mathcal{M} by randomly splitting each component of LC .
 - (b) Obtain model LC' by running multiple-restart EM [4] with \mathcal{M} as starting points.
 - (c) If $score(LC') > score(LC)$, set $LC := LC'$.

Figure 1: Algorithm *ParamReusing*

The algorithm *Standard* uses a standard starting point for EM. It has similar steps as *ParamReusing*. In step 2a, only one model M having one component more than LC is produced. The parameters of M are sampled from the uniform distribution. In step 2b, model LC' is obtained by running simple EM with M as a starting point.

²Here we assume that the log-likelihood term $LL(\mathbf{D}|LC)$ is part of the scoring function. The popular scoring functions such as BIC (or MDL), AIC, Cheeseman-Stutz, Laplace approximations [4] have this term.

The algorithm *StandardFixed* uses standard starting points for EM and estimates the model parameters when the number of components is fixed. It has similar steps as *ParamReusing*. In step 1, *LC* is a model with the specified number of components. In step 2a, each model in \mathcal{M} has the same number of components as *LC* and the parameters are sampled from the uniform distribution. Initially $|\mathcal{M}| = 1$ and in each iteration $|\mathcal{M}|$ is doubled.

3 Experimental Results

The setup of experiments is the following. As a model scoring function, we use the Cheeseman-Stutz score [3]. We use the maximum a posteriori (MAP) configuration of model parameters with uniform prior distributions, as described in [4], page 196. This setup has been reported by Chickering and Heckerman [4] to be more suitable than other scores for an LC model selection. We set the algorithm parameter p from Section 2.3 to 0.1. We run the EM algorithm until either the difference between successive values of log-likelihood is less than 0.01 or 200 iterations are reached. We run the experiments on a JavaTM 2 platform, 2.8 GHz Intel(R) processor.

First, we produced training data sets from the Reuters-21578, Distribution 1.0 text categorization test collection [14]. All the manifest variables are binary, and there are no missing values. Next, we obtained several classification data sets from the UCI Machine Learning Repository [1] and discarded the class information. Here the cardinality of manifest variables ranges from 2 to 10 and some data sets contain missing values.

For each data set, we run the algorithms 5 times, and the total running time for all the algorithms is the same. We stop iterating step 2 of the algorithms when the mean score for *Standard* stops to increase fast.³ The results are summarized in Table 1. The first 9 are text data sets, and the last 3 are from the UCI repository. For each data set, the following is shown: the number of instances, the number of manifest variables, the total running time (in minutes), the mean score (with a 95% confidence interval for a mean) and the average number of components of the final model for both algorithms, and the time ratio indicating how many times *ParamReusing* is faster than *Standard*.⁴ Bold text indicates a significant difference in score.

For data sets where the difference in score is significant, we run *StandardFixed*

³For each data set, we had that at stopping time t $\frac{score(t) - score(2t/3)}{score(2t/3) - score(t/3)} < 0.1$, where $score(u)$ is the mean score for *Standard* at time u .

⁴Time ratio is computed as $\frac{time_{Standard}}{time_{ParamReusing}}$, where $time_{Algorithm}$ is the time when the mean score of the model found by *Algorithm* is equal to the mean score of the final model found by *Standard* (for Crude20 data set, the final model found by *ParamReusing*).

Data set	Instan- ces	Vari- ables	Total time	<i>Standard</i>		<i>ParamReusing</i>		Time ratio
				Score	#C	Score	#C	
Corn10	10787	10	4	-10340.1 ± 1.7	8	-10339.1 ± 0.3	8	5.7
Crude10	10787	10	5	-10491.8 ± 0.7	6	-10491.3 ± 0.7	6	10.0
Earn10	10787	10	5	-26387.0 ± 7.1	11	-26380.7 ± 0.0	10	5.3
Corn20	10787	20	21	-23527.4 ± 2.4	10	-23527.2 ± 0.7	10	2.9
Crude20	10787	20	50	-23799.3 ± 1.0	9	-23799.4 ± 0.5	9	9.7
Earn20	10787	20	167	-46784.1 ± 13.5	15	-46767.2 ± 5.8	17	3.1
Corn30	10787	30	67	-40421.8 ± 8.2	11	-40409.9 ± 0.6	11	3.2
Crude30	10787	30	48	-28949.3 ± 10.3	9	-28943.4 ± 2.2	10	3.5
Earn30	10787	30	186	-74112.5 ± 71.4	14	-73922.3 ± 0.3	18	4.1
Solar	1389	10	1	-8343.2 ± 1.0	5	-8342.3 ± 0.5	5	2.3
Soybean	683	35	95	-11180.7 ± 29.2	14	-11072.4 ± 35.8	17	12.6
Mushroom	8124	22	280	-83858.7 ± 1400.2	15	-80935.0 ± 24.8	21	3.0

Table 1: Results for *Standard* and *ParamReusing*.

with the number of components being the same as in the best scoring model from *ParamReusing*. The results are summarized in Table 2.

Data set	<i>StandardFixed</i>		<i>ParamReusing</i> score	Time ratio
	Score	#C		
Corn30	-40431.2 ± 10.3	11	-40409.9 ± 0.6	5.4
Earn30	-73994.6 ± 48.9	18	-73922.3 ± 0.3	1.4
Soybean	-11113.5 ± 11.1	15	-11072.4 ± 35.8	5.0
Mushroom	-81796.7 ± 241.0	21	-80935.0 ± 24.8	2.4

Table 2: Results for *StandardFixed* and *ParamReusing*.

Figure 2 shows how the mean score changes during time for two data sets. The graphs for the other data sets look similar.

4 Discussion

In our experiments, parameter reusing was faster than the standard approach in all the tests. *ParamReusing* had a higher final mean score for almost all data sets (with the exception of Crude20), and that score was significantly higher for most data sets where the final models contained more than 10 components. Even when the standard approach was given an advantage of knowing the number of components (*StandardFixed*), *ParamReusing* was better on those many-component data sets. The most likely reason for such results is that the more components an LC model has, the lower the chances that a random starting point for EM will be a good starting point. Uebersax

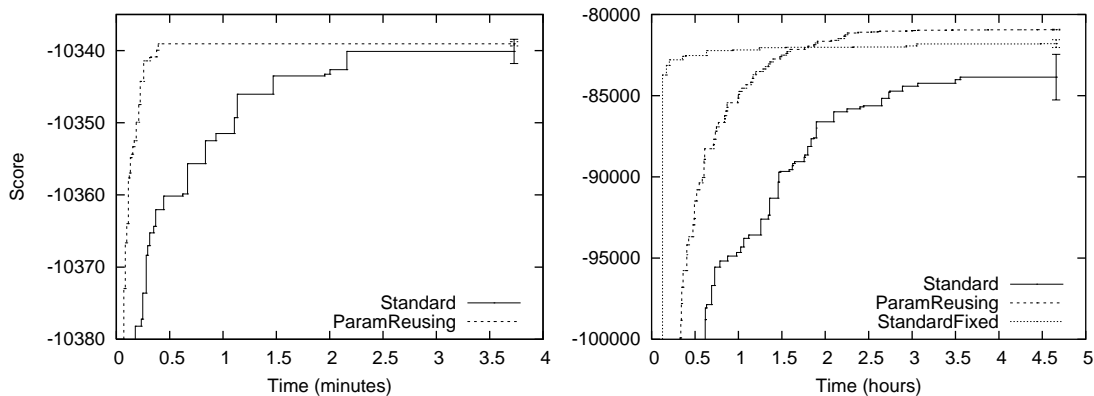


Figure 2: Score change during time for Corn10 (left) and Mushroom (right).

[18] also points out that the main factor contributing to local maximum solutions seems to be the number of components. By using the fact that LC models with different number of components have similar parameters, we are able to construct better starting points for EM.

Even though parameter reusing performs better than the standard approach, we have indications that *ParamReusing* does not find global maximum parameters and that it sometimes overestimates the number of components. That is why an operation for decreasing the number of components (for example, component merging) should be introduced. Another future work would be to extend the parameter reusing approach for learning the cardinalities of hidden variables in hierarchical latent class models [22] and in structural EM [10].

References

- [1] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [2] P. S. Bradley, U. M. Fayyad, and C. A. Reina. Scaling EM (expectation-maximization) clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, 1998.
- [3] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, 1995.
- [4] D. M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.

- [5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1977.
- [6] G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference*, pages 144–151, 2001.
- [7] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *AAAI-02*, pages 132–139, 2002.
- [8] U. M. Fayyad, C. A. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 194–198, 1998.
- [9] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.
- [10] N. Friedman. The Bayesian structural EM algorithm. In *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, pages 129–138, 1998.
- [11] Z. Ghahramani and M. J. Beal. Variational inference for Bayesian mixture of factor analysers. In *Advances in Neural Information Processing Systems 12*, pages 449–455, 2000.
- [12] L. A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61:215–231, 1974.
- [13] P. F. Lazarsfeld and N. W. Henry. *Latent structure analysis*. Boston: Houghton Mifflin, 1968.
- [14] David D. Lewis. Reuters-21578 text categorization test collection, 1997. <http://www.daviddlewis.com/resources/testcollections/reuters21578>.
- [15] C. Meek, B. Thiesson, and D. Heckerman. Staged mixture modelling and boosting. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighteenth Conference*, pages 335–343, 2002.
- [16] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Advances in Neural Information Processing Systems 11*, pages 543–549, 1999.
- [17] B. Thiesson, C. Meek, and D. Heckerman. Accelerating EM for large databases. Technical Report MSR-TR-99-31, Microsoft Research, 1999.
- [18] J. Uebersax. A brief study of local maxima solutions in latent class analysis, 2000. <http://ourworld.compuserve.com/homepages/jsuebersax/local.htm>.
- [19] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.
- [20] J. J. Verbeek, N. Vlassis, and B. Krose. Efficient greedy learning of Gaussian mixture models. *Neural Computation*, 15(2):469–485, 2003.
- [21] N. Vlassis and A. Likas. A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, 15(1):77–87, 2002.
- [22] Nevin L. Zhang. Hierarchical latent class models for cluster analysis. In *AAAI-02*, pages 230–237, 2002.