Learning Markov Networks with Bounded Inference Complexity

Ujjwal Das Gupta UJJWAL@UALBERTA.CA
Sriram Srinivasan SSRIRAM@UALBERTA.CA
Sanjeev Sharma SANJEEV1@UALBERTA.CA
Russell Greiner RGREINER@UALBERTA.CA
Department of Computing Science, University of Alberta

Abstract

In this paper, we study the problem of learning the structure of Markov Networks that permit efficient inference. We formulate structure learning as an optimization problem that maximizes the likelihood of the model such that the inference complexity on the resulting structure is bounded. The inference complexity is measured with respect to any chosen algorithm (either exact or approximate), or a distribution over any marginal or conditional query. We relate our work to previous approaches for learning bounded tree-width models and arithmetic circuits. The main contribution of our work is to isolate the inference penalty from the incremental structure building process. Our algorithm can be used to learn networks which bound the inference time of both exact and approximate algorithms. Further, we show that bounding inference time for approximate inference results in networks that exhibit less approximation error.

1. Introduction

Undirected probabilistic graphical models, such as Markov Random Fields (MRFs), are an important tool for modeling the joint probability density of a set of random variables. As the graph represents the independencies between the variables, it can be exploited to perform inference efficiently. These are extensively used in the domains such as, computer vision, robotics, computational biology, and natural language processing. Despite being a powerful tool, the application of Markov networks suffers from some problems.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

First, exact inference in MRFs is intractable in general, and the complexity of inference algorithms increases exponentially with the tree-width of the graph (Koller & Friedman, 2009). Approximate methods, such as Loopy Belief Propagation (Murphy et al., 1999), are widely used when the network is complex. However, the approximation error induced due to such methods is not well understood for arbitrary graphs. Second, learning the structure of an MRF from data is hard, especially when the knowledge about the dependencies between the variables is limited. An exponential number of undirected graphs can be constructed over a set of random variables. The best structure is commonly chosen based on a criterion similar to the Minimum Description Length (MDL) principle, which determines a trade-off between likelihood of the training data and the complexity of the model (Buntine, 1996). Complex models may fit the training data better, but they are prone to over-fitting. Also, inference on such models requires more time.

The structure learning problem can be formulated as a convex optimization problem of maximizing the likelihood of a completely connected Markov Network with an ℓ_1-regularization on the parameters of the model. This is a convex relaxation of the non-convex ℓ_0 penalty on the parameters imposed by the MDL. However, optimizing this criterion is still hard, as computing the gradient involves performing inference over the network, which is NP-hard in general. To solve this problem, one approach (Lee et al., 2006) used approximate inference techniques for computing the gradient, while using a greedy heuristic for choosing the order of parameters to activate during optimization. Although the objective is convex, the use of approximate inference makes the final objective value dependent on the order of activation. Another approach (Schmidt et al., 2008) used a pseudo-likelihood approximation of the likelihood, which does not involve inference calls for computing the gradient. As the sample size goes to infinity, the pseudo-likelihood estimates converge to likelihood estimates.

While ℓ_1-regularization methods reduce the risk of over-fitting, they do not guarantee efficient inference on the network, because the penalty drives a number of parameters to zero, regardless of the structure of the resulting graph. On complex graphs, approximate inference algorithms need to be used, which negate the benefit of learning a high likelihood model, as even an accurate model may result in inaccurate answers due to errors in approximate inference.

Methods that can learn structures permitting efficient exact inference have been studied previously. A tree structured graph allows polynomial time inference, and the best tree structured network can be learned efficiently (Chow & Liu, 1968). However, tree structured graphs are not expressive enough for many applications. In thin junction trees (Bach & Jordan, 2001), the inference complexity is penalized by bounding the tree-width of the triangulated network, and a heuristic algorithm is employed to construct the network. Another approach (Lowd & Domingos, 2008) for penalizing inference complexity is to directly learn an arithmetic circuit, which is equivalent to a Bayesian Network. A score for structure learning is formulated by penalizing the log likelihood by the number of edges in the arithmetic circuit, which is proportional to the complexity of inference in the arithmetic circuit. Sum product networks (Poon & Domingos, 2011) are another architecture that represent the marginal queries of a probability distribution compactly, and permit efficient exact inference.

All these approaches penalize or bound the complexity of inference as defined in a particular way. However, exact inference, using junction trees or arithmetic circuits, is not the only method for performing inference in an MRF. If approximate inference is going to be used over the resultant model, the tree-width or the arithmetic circuit size is not a good estimate of inference complexity. Further, the application may depend on the performance of a specific inference query, which is not adequately measured by these estimates.

We propose an algorithm that learns networks that permit fast inference, where the inference complexity can be specified as the time complexity of executing any combination of valid inference queries. The bound on the inference complexity can either be specified in terms of a complexity estimate that is specific to the algorithm used (for example, the maximum number of iterations during Loopy Belief Propagation), or, if such knowledge about the algorithm does not exist, the time used by the inference procedure can be bounded. We exploit the fact that optimizing the parameters of a MRF requires inference, and such an estimate can be made during this process. In case the application depends on a known distribution of queries, each of which is conditioned on a different set of variables, we can bound the complexity of such queries too, although this requires extra computation apart from the gradient calculation step. The incremental addition of structure in the Markov Network is combined with parameter optimization in the form of a stage-wise gradient descent algorithm, similar to the grafting approach to feature selection (Perkins et al., 2003).

Like the previous work on thin junction trees and arithmetic circuits, our approach is greedy and may find a locally-optimal solution. However, we generalize the principle behind these approaches to work with any inference procedure.

2. Preliminaries

Let $\mathcal{X} = \{X_1, X_2, ..., X_N\}$ be a set of discrete-valued random variables. The set of random variables can be represented as nodes in an MRF. The lack of an edge between two nodes denotes a pairwise independency between the two. MRFs can be represented using the framework of log-linear models. A log-linear model is represented using a set of feature functions $f_i(\mathcal{X})$, each of which is a function that returns a binary value for each assignment to some subset $\mathcal{X}_i \subset \mathcal{X}$. The model is parameterized by a vector of weights $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$, where each feature is associated with a weight θ_i. The corresponding Markov network has an edge between any two variables that appear together in some $\mathcal{X}_i$ corresponding to a feature. The probability of a particular assignment to $\mathcal{X}$ is given by

$$P(\mathcal{X}|\Theta) = \frac{1}{Z(\Theta)} \exp\left(\sum_{i=1}^{n} \theta_i f_i(\mathcal{X})\right)$$

where $Z(\Theta)$ is a partition function that normalizes the distribution.

2.1. Parameter Learning

The parameter learning for Markov Networks (MRFs) is posed as follows. Given a set of IID training instances $\mathcal{D} = \{x_1, x_2, .., x_m\}$, each being a complete assignment of values to variables, learn the parameters Θ that maximize the likelihood of $\mathcal{D}$. Equivalently, we can maximize the log-likelihood, given by

$$\ell(\Theta; \mathcal{D}) = \left(\sum_{j=1}^{m} \sum_{i=1}^{n} \theta_i f_i(x_j)\right) - m \log Z(\Theta)$$

An iterative procedure to maximize the log-likelihood requires computing the gradient, given by

$$\frac{1}{m}\frac{\partial \ell(\Theta;\mathcal{D})}{\partial \Theta_k} = \frac{1}{m}\sum_{j=1}^{m} f_k(x_j) - \mathbb{E}_{\Theta}[f_k(\mathcal{X})],$$

which is the difference between the observed and expected means of the features. To compute the expected feature counts, we must perform inference over the learned model.

3. Structure Learning with Bounded Inference Complexity

3.1. Optimization Problem Formulation

The problem of learning the best structure with a bound on inference complexity can be expressed as

$$\begin{array}{ll} \underset{\Theta}{\textbf{maximize}} & \ell(\Theta,\eta;\mathcal{D}) \\ \textbf{subject to} & R(\Theta,\eta) \le \tau \end{array}$$

where η is the structure of the underlying Markov Network, and $R(\Theta,\eta)$ is a function that represents the inference complexity over the chosen distribution of queries corresponding to the chosen inference algorithm. This is similar to a score based structure learning criterion like the Bayesian Information Criterion, with the model complexity penalty replaced by the inference complexity bound. Note that, while Θ is a vector of all possible parameters of the log linear model, only the ones corresponding to cliques present in η are active, and the others can be set to zero without changing the log likelihood expression. $R(\Theta,\eta)$ can be defined in many ways, including:

- Inference time (as measured by cpu clock cycles) of the inference algorithm. (This is a generic measure that can work with any algorithm.)
- Tree-width of the triangulated Markov network (This is an estimate of the exact inference time when using the Junction Tree algorithm, and results in a bounded tree-width model.)
- Number of iterations of the loopy-belief propagation algorithm. (This is an estimate of the inference time of LBP.)
- Number of iterations of the Tree-Reweighted Belief Propagation algorithm (TRBP) (Wainwright et al., 2003). (This is an estimate of the inference time of TRBP.)
- Size of the arithmetic circuit representing the network. (This is an estimate for the exact inference time after compiling the network to an arithmetic circuit (Darwiche, 2003), which is faster than the Junction Tree algorithm when there are many context specific independencies.)

While $\ell(\Theta,\eta)$ is a convex function of Θ, $R(\Theta,\eta)$ is non-convex and non-smooth under all of these definitions.

3.2. Algorithm

For the sake of simplicity, we first describe an algorithm which can learn pairwise Markov networks (MRFs restricted to two types of features: node features that depend only on a single random variable, and edge features that each depend on a specific pair of variables). Later, we show how this can be extended to general log-linear models, and how context specific independencies can be handled.

Algorithm 1 **Structure Learning for Pairwise Markov Networks with Bounded Inference Complexity**

```
Input: data $\mathcal{D}$ of size $m \times n$
$V \leftarrow \{1, 2, .., n\}$
$E \leftarrow \emptyset$
$\eta \leftarrow \{V, E\}$
$E_{cand} \leftarrow \{\langle$ u, v $\rangle | \forall$ u $\in$ V, v $\in$ V, u $\neq$ v $\}$
$\ell(\Theta,\eta;\mathcal{D})$ is the log likelihood of the data
Let $\Theta_E \subset \Theta$ be the parameters of edges E
Let $\Theta_V \subset \Theta$ be the parameters of nodes V
$\Theta_V \leftarrow \underset{\Theta_V}{argmax}\ \ell(\Theta,\eta;\mathcal{D})$
$\Theta_E \leftarrow 0$
repeat
  $e \leftarrow \underset{e \in E_{cand}}{argmax} \|\nabla_{\Theta_{\{e\}}} \ell(\Theta,\eta;\mathcal{D})\|_2$
  subject to $R(\Theta, \{V, E \cup \{e\}\}) \le \tau$
  $E \leftarrow E \cup \{e\}$
  $E_{cand} \leftarrow E_{cand} - \{e\}$
  $\{\Theta_V, \Theta_E\} \leftarrow \underset{\Theta_V,\Theta_E}{argmax}\ \ell(\Theta,\eta;\mathcal{D})$
until $e = \emptyset$
```

In Algorithm 1, the graph is initialized to an empty structure, and all possible edges are marked as inactive and added to the list of candidate edges. The network parameters Θ are divided into two sets, Θ_V and Θ_E, where Θ_V and Θ_E belong to the active set, corresponding to the node features and the features for the active edges, respectively. The remaining parameters are inactive and are incrementally introduced into the model. Similar to the grafting approach (Perkins et al., 2003), the features are introduced in a greedy manner, based on maximum increase of the objective function. Activating an edge activates multiple parameters in the model (one for each possible assignment to the pair of variables), and the increase of the objective with respect to an edge is measured as the ℓ_2 norm of the gradient with respect to the edge parameters.

As the algorithm needs to perform inference to compute the gradient, the evaluation of $R(\Theta, \eta)$ need not significantly alter the complexity of the algorithm. If the inference query for which R is defined is a part of the gradient computation step, R can be computed and stored in that step.

To extend this approach beyond pairwise models, we need to search among all candidate cliques that have not been added to the model, which are far more than the candidate edges alone. To reduce the search space, the hierarchical log-linear model can be used, in which a clique C may be added only if all cliques that are a subset of C are present in the model. A detailed analysis of how to optimize the active set for such a model has been conducted in past research (Schmidt & Murphy, 2010).

3.3. Regularization

We can add a regularization term like an ℓ_1 or ℓ_2 norm on the weight vector to the objective function without changing the algorithm. Adding an ℓ_1 norm would indeed lead to greater sparsity in the model. However, the constraint on $R(\Theta, \eta)$ acts as a regularizer by acting as the stopping condition for our greedy feature induction procedure. The grafting process has close connections to boosting, and the solutions obtained by boosting on each iteration are known to approximately follow the ℓ_1-regularized path (Rosset et al., 2004).

4. Experiments

To test the effectiveness of this algorithm, we ran experiments on learning a pairwise Markov Network from synthetic data. The graph was generated by randomly selecting each possible edge between 30 vertices with a probability of 0.5. A pairwise Markov Network was then generated by sampling the the feature weights from a standard normal distribution. This network was then sampled exactly to generate training and testing data, with 500 instances of each.

We compared our structure learning algorithm against a state-of-the-art ℓ_1-regularized structure learning algorithm (Schmidt et al., 2008). For comparison, two metrics were used: The Negative Log Likelihood (NLL) of the test data, and the Negative Conditional Marginal Log Likelihood (NCMLL) which has been used in the comparison of other structure learning algorithms as a measure of query specific performance (Lee et al., 2006).

For computing NCMLL, we divided the random variables into two groups x_{hidden} and $x_{observed}$. For each test instance $x[m]$, we compute $NCMLL = -\sum_{x \in x_{hidden}[m]} log(P(x|x_{observed}[m]))$. This is summed over all test instances, and averaged over 5 random partitions of the variables into the two groups.

Figures 1 to 4 show the comparisons of likelihood versus inference time for the two algorithms. Figures 1 and 2 use $R(\Theta, \eta)$ set to the tree-width. Figures 3 and 4 use $R(\Theta, \eta)$ equal to the inference time of the TRBP algorithm, where the inference queries under consideration is the set of all possible unconditional marginal queries of size 1 or 2 (that is, the set of all queries of the form $P(X_i)$ or $P(X_i, X_j)$, where X_i and X_j are random variables). In all cases, the likelihood is measured by exact inference, as it is a measure of the accuracy of the model. The y-axis represents inference time during testing. It is again measured as the time to compute all possible unconditional marginal queries of size 1 or 2, and we use the Junction Tree exact inference algorithm in Figure 1 and 2, and the TRBP algorithm in Figure 3 and 4.

It can be seen that in both our algorithm and ℓ_1-regularization, having a structure with higher inference time (a more complex structure) increases the likelihood of the test data, as a more accurate model is learned. However, this comes at the cost of inference complexity in both cases. If we compare two models in the graph, we see that Algorithm 1 can learn higher likelihood models, while having the same (or lower) inference cost as compared to ℓ_1-regularization.

Figure 5 shows the comparison of likelihood versus the approximation error of the TRBP algorithm. For Algorithm 1, $R(\Theta, \eta)$ is equal to the inference time of the TRBP algorithm. The approximation error is set to be equal to the percentage error of calculating $log(Z)$ using approximation inference. For both Algorithm 1 and ℓ_1-regularization, complex, higher likelihood models have higher approximation error, however Algorithm 1 outperforms ℓ_1-regularization for the same likelihood.

5. Discussion

We presented a generic algorithm for learning the structure under an inference complexity bound, and showed preliminary results that indicate that this can

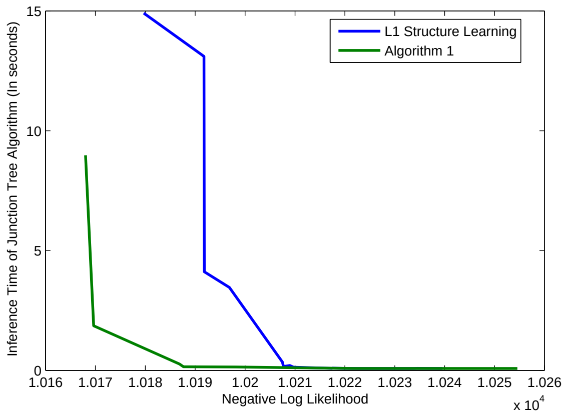

Figure 1. Negative log-likelihood of the test data vs. the inference time, for ℓ_1-regularized structure learning and our algorithm with $R(\Theta, \eta)$ set to tree-width

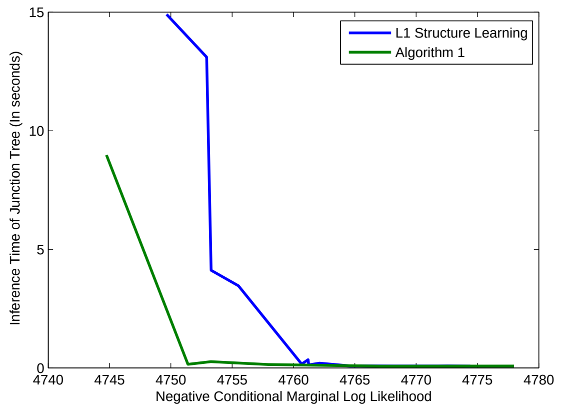

Figure 2. Negative conditional marginal log-likelihood of the test data vs. the inference time, for ℓ_1-regularized structure learning and our algorithm with $R(\Theta, \eta)$ set to tree-width

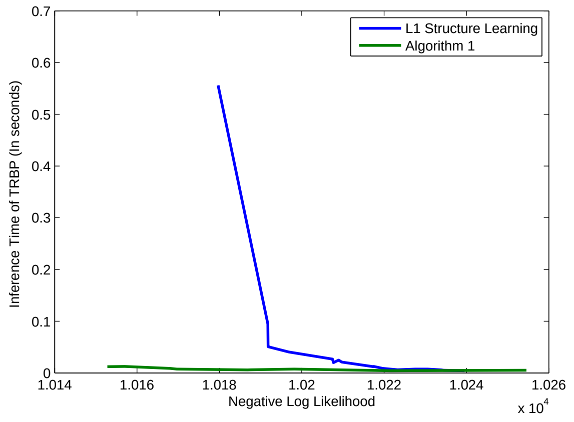

Figure 3. Negative log-likelihood of the test data vs. the inference time, for ℓ_1-regularized structure learning and our algorithm with $R(\Theta, \eta)$ set to inference time of the TRBP algorithm

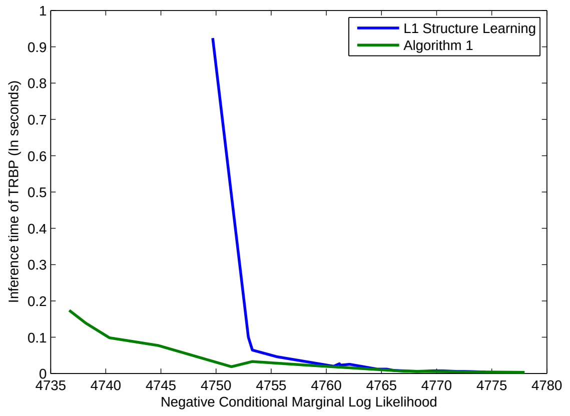

Figure 4. Negative conditional marginal log-likelihood of the test data vs. the inference time, for ℓ_1-regularized structure learning and our algorithm with $R(\Theta, \eta)$ set to inference time of the TRBP algorithm

learn higher likelihood and more accurate MRFs with a lower inference cost than the ℓ_1-regularized algorithms. We also showed that this approach can be used to learn bounded tree-width or low arithmetic circuit complexity networks under a unified setting.

By isolating the inference penalty and the feature induction mechanism, this algorithm allows many possible combinations to be made which have not been studied so far. Many feature selection mechanisms similar to Grafting exist (Zhu et al., 2010) and could be compared for their effectiveness in this setting.

To extend this approach to context specific independencies, we can activate parameters individually, rather than grouping them by edges. If an arithmetic circuit is going to be used to perform inference on such a model, there exist methods (Lowd & Domingos, 2008) to incrementally build the arithmetic circuit for computing $R(\Theta, \eta)$.

Our results show that bounding the inference time of approximation algorithms also results in lower approximation error, when compared to models of similar likelihood learned with ℓ_1-regularization. If low approximation error is the objective, instead of low inference time, it would be better to define $R(\Theta, \eta)$ as a direct measure of the approximation error. A simple way to do this would be to define $R(\Theta, \eta)$ as a cost function representing the difference between the value of the query under consideration, as computed by exact and approximate inference. A future research direction would be to develop a more sophisticated measure that does not require an exact inference computation step at each iteration of the algorithm.

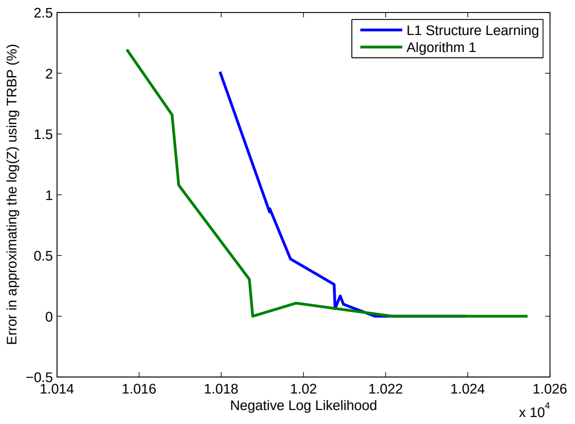

Figure 5. Negative log-likelihood of the test data vs. the error in approximating $\log(Z)$ with the TRBP algorithm, for ℓ_1-regularized structure learning and our algorithm with $R(\Theta, \eta)$ set to inference time of the TRBP algorithm

6. Acknowledgments

We would like to thank Ankush Roy and Sheehan Khan for their constructive feedback on our paper.

References

Bach, Francis R. and Jordan, Michael I. Thin junction trees. In *Advances in Neural Information Processing Systems 14*, pp. 569–576. MIT Press, 2001.

Buntine, Wray. A guide to the literature on learning probabilistic networks from data. *Knowledge and Data Engineering, IEEE Transactions on*, 8(2):195–210, 1996.

Chow, C and Liu, C. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

Darwiche, Adnan. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, May 2003. ISSN 0004-5411. doi: 10.1145/765568.765570.

Koller, Daphne and Friedman, Nir. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

Lee, Su-In, Ganapathi, Varun, and Koller, Daphne. Efficient structure learning of markov networks using l1regularization. In *In NIPS*. Citeseer, 2006.

Lowd, Daniel and Domingos, Pedro. Learning arithmetic circuits. In *UAI*, pp. 383–392, 2008.

Murphy, Kevin P, Weiss, Yair, and Jordan, Michael I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 467–475. Morgan Kaufmann Publishers Inc., 1999.

Perkins, Simon, Lacker, Kevin, and Theiler, James. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356, 2003.

Poon, Hoifung and Domingos, Pedro. Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 689–690. IEEE, 2011.

Rosset, Saharon, Zhu, Ji, and Hastie, Trevor. Boosting as a regularized path to a maximum margin classifier. *The Journal of Machine Learning Research*, 5: 941–973, 2004.

Schmidt, Mark and Murphy, Kevin. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Schmidt, Mark, Murphy, Kevin, Fung, Glenn, and Rosales, Rmer. Structure learning in random fields for heart motion abnormality detection. *CVPR. IEEE Computer Society*, 1:7, 2008.

Wainwright, Martin J, Jaakkola, Tommi S, and Willsky, Alan S. Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudomoment matching. In *Workshop on Artificial Intelligence and Statistics*, volume 21, pp. 97. Society for Artificial Intelligence and Statistics Np, 2003.

Zhu, Jun, Lao, Ni, and Xing, Eric P. Grafting-light: fast, incremental feature selection and structure learning of markov random fields. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 303–312. ACM, 2010.