Markov Logic Mixtures of Gaussian Processes: Towards Machines Reading Regression Data

Martin Schiegg[1,2], **Marion Neumann**[2], **Kristian Kersting**[2]
[1]Institute of Applied Information Processing
University of Ulm
89069 Ulm, Germany
martin.schiegg@alumni.uni-ulm.de

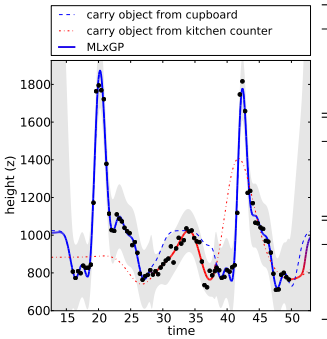
[2]Knowledge Discovery Department
Fraunhofer IAIS
53754 Sankt Augustin, Germany
{firstname.lastname}@iais.fraunhofer.de

Abstract

We propose a novel mixtures of Gaussian processes model in which the gating function is interconnected with a probabilistic logical model, in our case Markov logic networks. In this way, the resulting mixed graphical model, called *Markov logic mixtures of Gaussian processes* (MLxGP), solves joint Bayesian non-parametric regression and probabilistic relational inference tasks. In turn, MLxGP facilitates novel, interesting tasks such as regression based on logical constraints or drawing probabilistic logical conclusions about regression data, thus putting "machines reading regression data" in reach.

1 Introduction

Can computers learn to read regression data? Imagine you want to get a declarative description of your possibly high-dimensional dataset containing a huge amount of observations. That is, next to providing a regression model, your computer reads, i.e. interprets your regression data for you in terms of a declarative probabilistic knowledge base. Indeed, this is akin to Machine Reading (MR), the automatic unsupervised understanding of text. According to Etzioni et al. [10] MR combines information extraction and reasoning to draw conclusions about implicitly given knowledge. In analogy, we investigate the problem of machines reading regression data (MR^{2}D). That is, given a regression dataset together with weighted logical knowledge, is it possible to automatically understand and thus be able to "read" the regression data?



Weight	First-Order Logic
1.0	segment(O,T) ⇔ segment(O,T+1)
∞	onT(O,T) ⇒ onT(O,T+1)
⋮	⋮
10.0	segment(O!,T)

Evidence		
inCb(p)	¬onT(p,18.4)	¬onT(p,25.2)
inCb(c)	¬onT(c,18.4)	onT(p,25.7)
¬inCb(k)	atT(35.7)	onT(k,43.3)
¬onT(k,18.4)	atT(25.2)	onT(c,43.3)

Subset of Inferred Beliefs	
onT(k,22.4)	0.0
onT(k,44.8)	1.0
segment(k,32.5)	1.0
segment(c,41.6)	1.0

Figure 1: **Kitchen Regression Data (left) and Logical Knowledge (right)** While setting the table, the height of the subject's right hand is observed over time (left). Background knowledge about the location of objects and rules describing actions to set the table are given (right). `onT(O)` short for `onTable(O,T)` indicates whether object `O` is located on the table at time `T`, `inCb(O,T)` short for `inCupboard(O)` defines the storing location of `O`, and `atT(T)` short for `atTable(T)` represents whether the subject is at the table at time `T`. Objects are `plate` (`p`), `cup` (`c`), and `knife` (`k`), and `segment(O,T)` indicates whether the movement at time `T` is assigned to the manipulation of object `O`.

As an example consider the situation depicted in Fig. 1. It shows a regression dataset as well as relational background knowledge extracted from the real-world TUM Kitchen Dataset [26]. A subject is setting the table and the height of the subject's right hand is observed over time. Additionally, background knowledge about the location of objects and rules describing actions to set the table are given. By combining both types of information, the regression data and the declarative background knowledge, can a machine actually understand the process of setting the table?
Other situations in which both regression data and weighted logical knowledge is available are, for instance, GPS tracks enriched with points of interest

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume 22 of JMLR: W&CP 22.

or social network information, and climate data combined with complex knowledge about emission pathways. Further examples can be found in robotics where binary and continuous knowledge gained from different sensors together with logical background knowledge can be used to improve context-specific grasping, terrain modeling or robot localization, among others.

In this context, our main contribution is a novel model that intertwines Gaussian process (GP) regression [20] within probabilistic logical languages. Whereas GPs capture the regression data well, probabilistic logical languages, see e.g. [11, 7, 6] for overviews, provide powerful formalisms to compactly represent noisy declarative knowledge. Specifically, we introduce Markov logic mixtures of Gaussian processes (MLxGP). It takes Tresp's [27] mixtures of Gaussian processes (MGP) model and intertwines the gating function with a probabilistic relational model – in our case Richardson and Domingos' [21] Markov logic networks (MLNs)[1]. In doing so, MLxGP facilitates joint Bayesian non-parametric regression and probabilistic logical inference. Thus, MLxGP puts MR^{2}D in reach.

Reconsider the Kitchen example in Fig. 1. Here, an MLxGP can exploit two sources of knowledge, the MLN and the MGP, to achieve a segmentation of the data (as indicated by the colors in Fig. 1) and, in turn, an understanding of the observed data in terms of the declarative background knowledge. Actually, MLxGP reads the data as follows: *"To set the table the subject first carried the plate, then the knife, followed by the cup."* Indeed, one may argue that inference in the GP and the probabilistic logical model separately is sufficient to achieve the same reading. However, as shown by our experiments, this is not the case, at least without expensive feature engineering. The probabilistic logical model and in turn the understanding of the regression data is improved by knowledge captured in the GP. The GP regression benefits from the probabilistic logical model since it influences predictions for missing values.

We proceed as follows. We start off by touching upon related work and briefly reviewing (M)GPs and MLNs. Then, we introduce MLxGP and describe how to perform inference and how to learn the mixture weights and the hyperparameters. Before concluding, we present our experimental evaluation.

2 Related Work

MLxGPs combine two lines of research: Gaussian processes, in particular mixtures of Gaussian processes, and statistical relational learning.

A multitude of MGPs have been proposed in literature [27, 19, 14, 29, 1, 25, 22]. We adapt the idea of Tresp [27] that each observation comes from one individual unobserved GP, termed *expert*. From the observations, expert mixture weights are learned. Experts obtain infinite noise terms for observations if they do not belong to this expert. However, in contrast to classical MGPs, MLxGP declaratively "explains" both the experts and the weights by modeling the dependencies between the experts as well as between data points by a probabilistic logical model. Deisenroth et al. [8] proposed the idea that a classifier selects one out of two GPs to predict a target value for a test input. Again, this approach does not handle implicit knowledge derived from a probabilistic logical knowledge base (KB). Indeed, several relational GPs have been proposed, see e.g. [5, 23, 30, 17]. However, they are no mixtures and assume relations between data points to be extensional, i.e., given as a set of true and false ground facts rather than intensional, i.e., in terms of rules as in MLxGP.

Finally, within the statistical relational learning community, hybrid models incorporating continuous variables into probabilistic logical models have been proposed, see e.g. [28, 12, 4]. Also, there has been extensive work on constructing process model from continuous data using declarative background knowledge, see e.g. [3] and reference in there. However, these approaches are all of parametric nature although Bayesian non-parametric methods are desirable to model non-linear functions, for instance.

3 Mixtures of Gaussian Processes

Gaussian processes (GP) [20] are an extremely attractive method to perform non-linear regression because of their flexible non-parametric nature, their analytical properties and the predictive uncertainties they provide. Assume we have a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n} = \{(X, \mathbf{y})\}$ of n multi-dimensional inputs and their corresponding scalar outputs. The regression task is, given a new input $\mathbf{x}_*$, to obtain the predictive distribution for the corresponding observation y_*. The output $y_i = f(\mathbf{x}_i) + \varepsilon_i$ is assumed to be generated by an underlying latent function and additive independent Gaussian noise $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Now, a GP prior on $f(\mathbf{x}_i) \sim \mathcal{GP}(\mathbf{0}, k(\mathbf{x}_i, \mathbf{x}_j))$ yields a multivariate Gaussian distribution for $\mathbf{f} = f(X) = (f(\mathbf{x}_1), ..., f(\mathbf{x}_n))^\top$ with zero mean and covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$, which may be any Mercer kernel function with hyperparameters $\boldsymbol{\theta}$. For a new input case $\mathbf{x}_*$, the predictive distribution is Gaussian and given by $f_* \mid X, \mathbf{y}, \mathbf{x}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*))$. For the sake

[1] Other probabilistic relational models could be used instead, as long as they induce a factor graph.

of clarity, we focus on predictions for one test input. See [20] for a comprehensive introduction to GPs.

MGPs are a variant of the mixture of experts model [13], which extend a single GP to handle multiple latent functions. Each individual expert $\mathcal{GP}_k \in \{\mathcal{GP}_1, ..., \mathcal{GP}_R\}$ locally represents the dependencies in the data by using respective hyperparameters $\Theta = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_R\}$. Fig. 2 (upper part) graphically illustrates such an MGP model for regression using the gates notation [15] to indicate that the Gaussian field of latent variables f_i is indeed a mixture of GP experts. Dependent on the state of the nominal *selector variable* z connected to the gate itself, GP mixtures are created. The probability distribution of the output y_* is determined by

$$P(y_* \mid \mathbf{x}_*, \Theta, X, \mathbf{y}) = \sum_{z_*} P(y_*, \mathbf{z}_* \mid \mathbf{x}_*, \Theta, X, \mathbf{y}) = \sum_{k=1}^{R} P(z_* = k \mid \mathbf{x}_*)\, \mathcal{N}(y_*; \bar{f}_*^{\,k}, \mathrm{cov}(f_*^k)),$$

where $\mathcal{N}(y_*; \bar{f}_*^{\,k}, \mathrm{cov}(f_*^k))$ is the Gaussian density of y_* calculated at input $\mathbf{x}_*$ given mean and covariance of $\mathcal{GP}_k$ and $P(z_* = k \mid \mathbf{x}_*, \mathbf{z})$ is the probability that y_* is generated by the k-th component of the mixture.

4 Markov Logic Networks

First-order probabilistic models essentially join graphical models with elements of first-order logic by defining template factors (such as Poole's parfactors [18]) that apply to whole sets of objects at once. A simple and powerful such language is Markov logic [21]. A Markov logic network (MLN) consists of a set of formulas F_i in first-order logic weighted by $\omega_i \in \mathbb{R}$. By plugging in a finite set of constants into the formulas, the *ground* MLN results in a Markov network which contains one binary node for each possible grounding of a predicate and one feature per formula. A Markov network is called to be a *world* if there are values assigned to all the nodes and features. Then, the probability distribution of a world x is given by the log-linear model

$$P(x) = \frac{1}{Z} \exp\left(\sum_i \omega_i n_i(x)\right) \tag{1}$$

where $n_i(x)$ is the number of true groundings of formula F_i in the considered world x and Z is the partition function. Eq. (1) is equivalent to the product of potential functions $P(x) = \frac{1}{Z}\prod_i \phi_i\,(x_i)^{n_i(x)}$, where ϕ_i is the potential for formula F_i. A world is not impossible if a single formula is violated as it would be the case in first-order logic. However, it is less likely, dependent on the relevance of the formula which is determined by its weight. An important inference task in MLNs is to compute the conditional probability of variables given the truth values of some others, the evidence E, by marginalizing over the remaining variables. Belief propagation [16], for instance, efficiently solves this problem. For the rest of this paper, we denote the information provided by the MLN by $\eta = \{F, \boldsymbol{\omega}, E\}$. Tab. 1 shows an MLN for the Kitchen example. Following logic programming convention, arguments in upper case denote variables/placeholders, e.g. `inCupboard(O)`, and in lower case ground predicates, e.g. `inCupboard(plate)`. The bottom of Fig. 2 shows part of the induced graphical model.

Table 1: **The Kitchen MLN** The mutual exclusive predicates (denoted by `O!`) `expert(1,T)` and `expert(2,T)` indicate whether an object is carried from the cupboard respectively the kitchen counter. Variables are typed such that `T` represents points in time, `O` stands for objects, and `E` ranges over the experts. `T+1` denotes the successive time step of `T`.

Rule	First-Order Logic	Weight
Only one object is carried at once.	`segment(O!,T)`	10.0
If an object is stored in the cupboard, the movements to carry it are modeled by expert 1.	`(segment(O,T)` ∧ `inCupboard(O))` ⇔ `expert(1,T)`	10.0
... otherwise by expert 2.	`(segment(O,T)` ∧ ¬`inCupboard(O))` ⇔ `expert(2,T)`	10.0
One object is carried in successive points in time.	`segment(O,T)` ⇔ `segment(O,T+1)`	1.0
An object remains on the table.	`onTable(O,T)` ⇒ `onTable(O,T+1)`	∞
An object, carried in one instant of time and not carried in the successive point in time, is being placed on the table.	`((segment(O,T)` ∧ ¬`segment(O,T+1))` ⇔ `(onTable(O,T+1)` ∧ ¬`onTable(O,T)))`	40.0
If the subject is at the table, the carried object is put there at that moment.	`(segment(O,T)` ∧ `atTable(T)` ⇒ `(`¬`onTable(O,T)` ∧ `onTable(O,T+1)))`	40.0
Usually, the subject is not at the table.	¬`atTable(T)`	3.0
An object located on the table is not carried at the same time.	`onTable(O,T)` ⇒ ¬`segment(O,T)`	4.0
Exactly one expert is active at the same time.	`expert(E!,T)`	∞

5 Markov Logic Mixture of GPs

Experts and mixture weights of the MGP approach are based on training regression data only. Markov logic mixture of GPs (MLxGP) extends this idea by also conditioning the mixture weights on an MLN.

5.1 The Model

The MLxGP model consists of two parts, the mixture of experts model and an MLN. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,...,n}$ be a regression dataset, e.g. motion

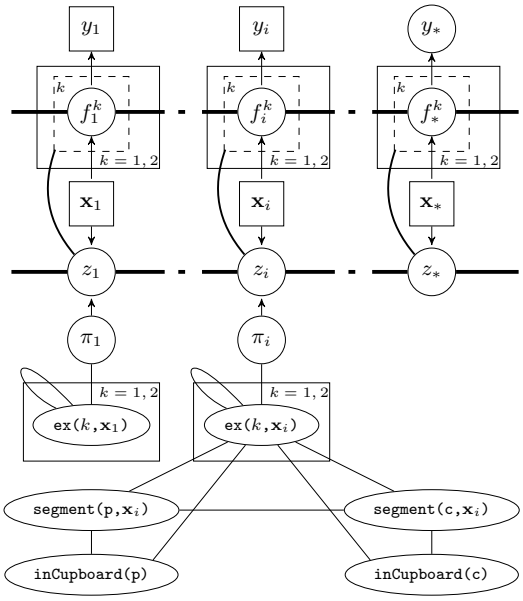


Figure 2: **Mixed Graphical Model for MLxGP** The upper part is a chain graph for a mixture of two GPs for regression. Squares represent observed variables and circles unknowns. Dashed rectangles illustrate gates and solid lined rectangles plates. The thick horizontal bar represents a set of fully connected nodes. Note that an observation y_i is conditionally independent of all other nodes given the corresponding latent variable f_i^k. The lower part is the ground graphical model for the Kitchen example applying the first three rules and the last rule of Tab. 1 to the constants `plate` (`p`) and `cup` (`c`). $\{(\mathbf{x}_i, y_i)\}_{i=1,...,n}$ denotes the regression dataset, z_i represent the selector variables. The node `expert(`k`,`$\mathbf{x}_i$`)` (`ex(`k`,`$\mathbf{x}_i$`)`), indicates whether training case $\mathbf{x}_i$ belongs to expert $\mathcal{GP}_k$.

capture data as in Fig. 1, and $\eta = \{F, \boldsymbol{\omega}, E\}$ is the information provided by the knowledge base consisting of first-order formulas F_j which might be dependent on the input of the regression data, their assigned weights ω_j, and some evidence E. The goal is to learn both the hyperparameters $\Theta = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_R\}$ of the GP experts $\mathcal{GP}_1, ..., \mathcal{GP}_R$, where $\boldsymbol{\theta}_k = (\theta_1^k, ..., \theta_{m_k}^k)^\top$, $m_k = |\boldsymbol{\theta}_k|$ and $k \in \{1, 2, ..., R\}$ and their mixture weights in order to predict unknown output values as well as truth values of ground atoms. Let $\mathbf{z}$ be the latent categorical *selector variable* indicating which expert is active for the corresponding input, then the probability $P(z_i = k)$ of y_i being generated by the k-th expert $\mathcal{GP}_k$ serves as a mixture weight. For each expert $\mathcal{GP}_k$ and training input $\mathbf{x}_i$, the MLN contains an atom `expert(`k`,`$\mathbf{x}_i$`)` which is connected to the corresponding selector variable z_i. An illustrating mixed graphical model of the MLxGP model considering an excerpt of the Kitchen example is shown in Fig. 2.

The general idea of the MLxGP model is that both the

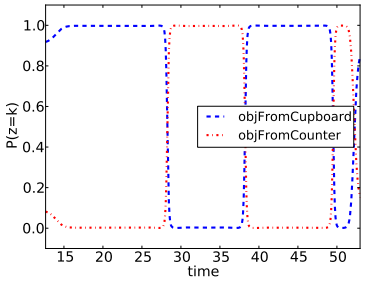


Figure 3: **Kitchen Gating Function** The gating GPs provide a segmentation of the input space. In the Kitchen example, they indicate whether an object is carried from the cupboard or the kitchen counter.

estimates about the mixture weights provided by the MLN, which can be interpreted as a declarative prior for the MLxGP, and the noise of the predictions of the experts indicate how well a data point is explained by a specific expert. In MLxGPs, we model this property by an artificial input-dependent noise constituting a soft assignment of training points to GP experts. If, in the extreme case, the data point $(\mathbf{x}_i, y_i)$ is not included in the training set of expert $\mathcal{GP}_k$, then the noise term of $\mathcal{GP}_k$ is set to a high value, up to infinity. If, on the other hand, $(\mathbf{x}_i, y_i)$ is assigned to $\mathcal{GP}_k$, then the artificial noise term is close to zero.

The predictive mean and variance of expert $\mathcal{GP}_k$ are

$$\bar{f}_i^{\,k} = K^k(\mathbf{x}_i, X)[K^k(X, X) + \Psi^k + \sigma_k^2 I]^{-1}\mathbf{y}, \quad (2)$$

$$\begin{aligned}\mathrm{cov}(f_i^k) =& K^k(\mathbf{x}_i, \mathbf{x}_i) + \sigma_k^2 - K^k(\mathbf{x}_i, X)\cdot \\ & [K^k(X, X) + \Psi^k + \sigma_k^2 I]^{-1} K^k(X, \mathbf{x}_i),\end{aligned} \quad (3)$$

where the diagonal matrix Ψ^k constitutes the input-dependent *artificial noise* term and σ_k^2 is the noise variance of expert $\mathcal{GP}_k$. Both terms are obtained from the learning phase of the MLxGP.

5.2 Inference

In order to do joint inference within the MGP and the MLN, we have to introduce a gating function.

Gating Function: Given the estimated probabilities of the selector variables $w_i^k = P(z_i = k \mid X, \mathbf{y}, \eta)$, and the artificial noise terms Ψ^k (both will be introduced later), and given the hyperparameters Θ of the GP experts, the key component is the *gating function*. It defines the mixture assignment at an unknown input case $\mathbf{x}_*$ based on the latent selector variable z_* which is influenced by both the KB and the GP. As gating function for the MLxGP model, we introduce an R-class classification Gaussian process model determining the probability of data point $\mathbf{x}_*$ to be generated by expert

$\mathcal{GP}_k$: $w_*^k =$

$$P(z_* = k \mid \mathbf{x}_*, X, \mathbf{w}^1, ..., \mathbf{w}^R) = \frac{\exp\left(r \bar{f}_*^{z,k}\right)}{\sum_{\nu=1}^{R} \exp\left(r \bar{f}_*^{z,\nu}\right)}, \quad (4)$$

where $\bar{f}_*^{z,k}$ is the prediction of $\mathcal{GP}_k^z$ with a shared output scale r. $\{\mathcal{GP}_1^z, ..., \mathcal{GP}_R^z\}$ is a second set of so-called gating GPs with hyperparameters $\Theta^z = \{\boldsymbol{\theta}_1^z, ..., \boldsymbol{\theta}_R^z\}$, where $\boldsymbol{\theta}_k^z = (\theta_1^{k,z}, ..., \theta_{s_k}^{k,z})^\top$, $s_k = |\boldsymbol{\theta}_k^z|$ and $k \in \{1, 2, ..., R\}$. In our running example, the two gating GPs illustrated in Fig. 3 indicate the probability of the subject carrying an object from the cupboard or the kitchen counter at an arbitrary point in time. The *rescaling factor* r controls the hardness of the assignments of unknown data points to the expert GPs. The larger r, the harder the assignment to one single GP expert. For reasons of model complexity, we do not learn r, but fix it to a value of 6 in all experiments.

To learn the gating GPs, the selector variables are treated as noise-free observations and the training set $\{(\mathbf{x}_i, w_i^k)\}_{i=1,...,n}$ is used to train a standard GP $\mathcal{GP}_k^z$ for each k. The belief $P(z_{\text{dummy}} = k \mid X, \eta)$ obtained from the KB is used as a constant mean function for $\mathcal{GP}_k^z$ and contains the information provided by the KB for all data points not contained in $\mathcal{D}$. Technically, we have to add a constant $\mathbf{x}_{\text{dummy}}$ to the KB which covers all test cases for which no regression data is given, e.g. $\mathbf{x}_{\text{dummy}}$ covers all points in time with missing measurements of motion capture data.

Predictive Distribution and Beliefs: Given the gating function, we can now compute predictive distributions and beliefs. Thus, it is possible to predict continuous regression values based on the gating function as well as truth values for ground atoms in the knowledge base. The predictive distribution corresponding to a new input case $\mathbf{x}_*$ has mean

$$\bar{f}_* = P(y_* \mid \mathbf{x}_*, X, \mathbf{y}, \mathbf{z}, \Theta) = \sum\nolimits_{k=1}^{R} w_*^k \bar{f}_*^k,$$

where w_*^k is determined by Eq. (4), and $\bar{f}_*^k$ is the predictive mean at test input $\mathbf{x}_*$ of expert $\mathcal{GP}_k$ given by Eq. (2). The predictive variance is determined by

$$\operatorname{cov}(f_*) = \sum_{k=1}^{R} w_*^k \left(\operatorname{cov}(f_*^k) + \left(\bar{f}_*^k\right)^2\right) - \left(\sum_{k=1}^{R} w_*^k \bar{f}_*^k\right)^2,$$

where $\operatorname{cov}(f_*^k)$ is computed as in Eq. (3). The resulting mixture, denoted as MLxGP, is a Gaussian Process.

Inference in the KB can be performed by any inference algorithm such as message passing techniques like belief propagation or junction tree.

For the Kitchen MLxGP as depicted in Fig. 2 interesting predictions are the height of the subject's right hand while grasping a specific object, or which object (cup, plate or knife) is carried at which time. These predictions provide sensible information to enable the understanding of the regression curve of the MLxGP.

5.3 Learning

The learning phase of the MLxGP consists of two steps, first we estimate the mixture weights and then we learn the final hyperparameters of the GP experts.

Estimating Mixture Weights: To estimate the mixture weights (learning the MLN is left as future work), we learn them simultaneously with the hyperparameters of the GP experts. Hence, we maximize the expected log-likelihood for both $\mathbf{y}$ and $\mathbf{z}$

$$\begin{aligned} Q(X, \mathbf{y}, \Theta, \mathbf{z}, \eta) &= \mathbb{E}\left(\log P(\mathbf{y}, \mathbf{z} \mid X, \Theta, \eta)\right) \\ &= \sum_{i=1}^{n} \sum_{k=1}^{R} w_i^k \left(\log P\left(y_i \mid X, \mathbf{y}_{-i}, \boldsymbol{\theta}_k, \eta, z_i = k\right)\right. \\ &\quad \left. + \log P\left(z_i = k \mid X, \mathbf{y}_{-i}, \Theta, \eta\right)\right), \end{aligned} \quad (5)$$

where $\mathbf{y}_{-i}$ denotes all training outputs except y_i and w_i^k is the mixture weight for $\mathbf{x}_i$ of expert $\mathcal{GP}_k$. Since MLN and GP are separated by the the selector variables z_i as depicted in Fig. 2, we apply the Expectation-Maximization algorithm [9].

In the **E-step**, we compute the probability $w_i^k = P(z_i = k \mid X, \mathbf{y}, \eta)$ of the selector variable, based on the GP hyperparameters $\Theta^{(t-1)}$ obtained from the previous M-step. To do so, we first compute a prior for the MLN atoms `expert(`$k, \mathbf{x}_i$`)` processing information from the GP experts model, i.e.

$$P(z_i = k \mid X, \mathbf{y}, \boldsymbol{\theta}_k^{(t-1)}) = \frac{\mathcal{N}\left(y_i; \bar{f}_i^k, \operatorname{cov}(f_i^k)\right)}{\sum_{\nu=1}^{R} \mathcal{N}\left(y_i; \bar{f}_i^\nu, \operatorname{cov}(f_i^\nu)\right)},$$

where $\mathcal{N}(y; m, \sigma^2)$ is the likelihood of y given mean m and variance σ^2. To incorporate this information into the MLN, let $g_{i,k}(s)$ be the feature of the clique only containing the `expert(`$k, \mathbf{x}_i$`)` node, i.e. the feature corresponding to the unary clique potential of this node, with states $s \in \{0, 1\}$. We set this feature to the current guess of whether data point $\mathbf{x}_i$ belongs to expert $\mathcal{GP}_k$ or not, i.e.

$$g_{i,k}(s) = \begin{cases} P(z_i = k \mid X, \mathbf{y}, \boldsymbol{\theta}_k^{(t-1)}), & \text{for } s = 1, \\ 1 - P(z_i = k \mid X, \mathbf{y}, \boldsymbol{\theta}_k^{(t-1)}), & \text{for } s = 0. \end{cases}$$

Using standard inference techniques for the MLN provides the belief $\pi_i^k = P(z_i = k \mid X, \eta)$. Combining π_i^k and the likelihood yields the normalized expert mixture weights $w_i^k = P(z_i = k \mid X, \mathbf{y}, \theta_k^{(t-1)}, \eta) =$

$$\frac{\pi_i^k \mathcal{N}(y_i; \bar{f}_i{}^k, \operatorname{cov}(f_i^k))}{\sum_{\nu=1}^{R} \pi_i^\nu \mathcal{N}(y_i; \bar{f}_i{}^\nu, \operatorname{cov}(f_i^\nu))}.$$

In the **M-step**, based on the estimates w_i^k and π_i^k of the E-step, the hyperparameters $\Theta^{(t)}$ for the GP experts are learned. We therefore maximize the objective function, Eq. (5), w.r.t. the hyperparameters $\Theta^{(t)}$ of the expert GPs using, for instance, gradient based optimization. Unfortunately, $\log P(z_i = k \mid X, \mathbf{y}_{-i}, \Theta^{(t)}, \eta)$ in Eq. (5) has no closed form and $\Theta^{(t)}$ is unknown. Therefore, differentiation w.r.t. Θ is intractable and we approximate this term by $\log P(z_i = k \mid X, \mathbf{y}_{-i}, \Theta^{(t-1)}, \eta)$ using the hyperparameters $\Theta^{(t-1)}$ from the previous iteration. This yields the approximate objective function $\tilde{Q}$.

We set the covariance matrix of expert $\mathcal{GP}_k$ as $K^k = K^k(X, X) + \Psi^k + \sigma^2 I$, where σ^2 is the noise variance obtained from a standard GP trained on the entire regression dataset and Ψ^k is a diagonal matrix constituting the *artificial noise* featuring a soft assignment of training points being considered for prediction for expert $\mathcal{GP}_k$. ψ_{ii}^k intuitively consists of two terms:
(a) Contribution from MLN: Based on the estimates π_i^k determined from the MLN, the term $\frac{1-\pi_i^k}{\pi_i^k}$ obtains a value close to zero if the current belief is that the training point $(\mathbf{x}_i, y_i)$ belongs to expert $\mathcal{GP}_k$, and a value up to infinity otherwise.
(b) Contribution from GP: $\frac{1}{n}\sum_{j=1}^n \pi_j^k (\bar{f}_j^k - y_j)^2$ indicates the estimate for the current noise of expert $\mathcal{GP}_k$ and is zero for a perfect prediction of all training cases and positive otherwise. This term guarantees that during weight learning, predictions for actual training cases of the respective expert are close to their observed values.
In order to preserve the idea of the artificial noise ranging from zero to infinity dependent on how probable the corresponding data point belongs to the training set of the considered expert, these two terms are multiplied. Hence, the artificial noise

$$\psi_{ii}^k = \frac{\left(1-\pi_i^k\right)\left(\sum_{j=1}^n \pi_j^k \left(\bar{f}_j^k - y_j\right)^2\right)}{n\pi_i^k}$$

ranges from zero to infinity, whereas ψ_{ii}^k is zero if $(\mathbf{x}_i, y_i)$ is generated by the k-th expert, i.e. $w_i^k = 1$, and takes an infinite value if not, i.e. $w_i^k = 0$. In the latter case, the i-th row and i-th column of the inverse covariance matrices in Eqs. (2) and (3) are zero. In turn, observation y_i has no influence on the prediction of expert $\mathcal{GP}_k$ since it is treated as not being part of its training set.
Now, by computing

$$\frac{\partial \bar{f}_i^k}{\partial \theta_j^l} = \begin{cases} \frac{[Z_j^l \boldsymbol{\alpha}^l]_i}{[(K^l)^{-1}]_{ii}} - \frac{\alpha_i^l [Z_j^l (K^l)^{-1}]_{ii}}{[(K^l)^{-1}]_{ii}^2}, & k = l \\ 0, & k \neq l \end{cases},$$

$$\frac{\partial \mathrm{cov}(f_i^k)}{\partial \theta_j^l} = \begin{cases} \frac{[Z_j^l (K^l)^{-1}]_{ii}}{[(K^l)^{-1}]_{ii}^2}, & k = l \\ 0, & k \neq l \end{cases}$$

Table 2: **MLxGP vs. GPR**

Dataset	*NRMSE*		*NLPD*	
	GPR	MLxGP	GPR	MLxGP
Kitchen	**0.04**	0.05	5.88	**5.77**
Terrain	0.10	**0.04**	3.57	**0.22**
Friends-Coffee	0.28	**0.12**	**3.34**	10.11
Motorcycle	0.08	0.08	4.58	**4.28**

Table 3: **MLxGP vs. MLN**

Dataset	*CMLL*	
	MLN	MLxGP
Kitchen	−57.81	**-1.07**
Terrain	−14.13	**-9.16**
Friends-Coffee	−18.20	**-2.96**
Motorcycle	−0.01	−0.01

with $\boldsymbol{\alpha}^l = (K^l)^{-1}\mathbf{y}$ and $Z_j^l = (K^l)^{-1}\frac{\partial K^l}{\partial \theta_j^l}$, the partial derivatives of the objective function $\tilde{Q}$ w.r.t. the experts' hyperparameters are given by

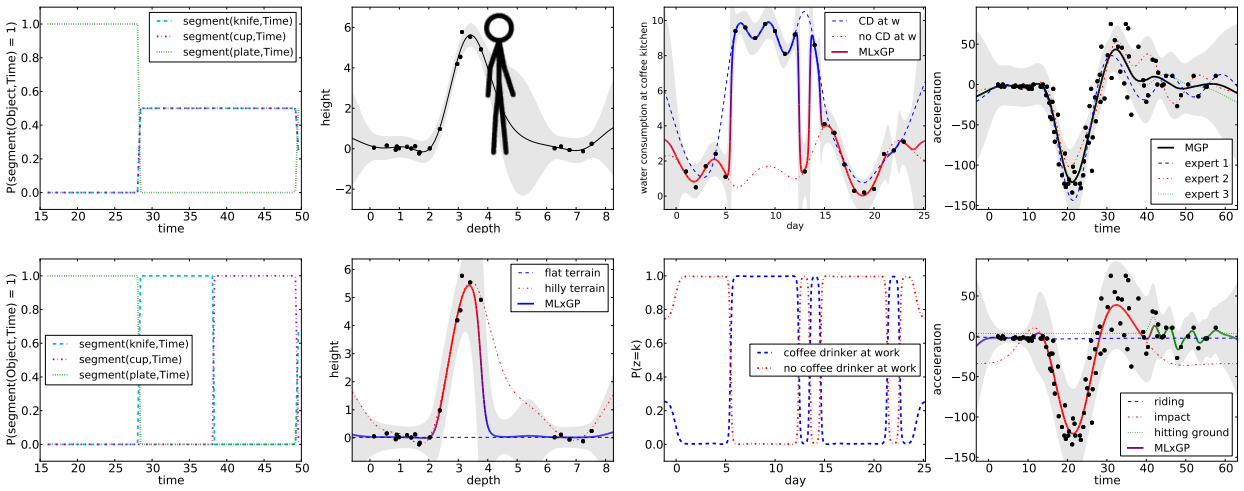
$$\begin{aligned}\frac{\partial \tilde{Q}^{(t)}}{\partial \theta_j^l} =& \sum_{i=1}^n \sum_{k=1}^R w_i^k \left(\frac{\partial \log P(y_i \mid X, \mathbf{y}_{-i}, \Theta^{(t)}, \eta, z_i = k)}{\partial \bar{f}_i^k} \frac{\partial \bar{f}_i^k}{\partial \theta_j^l} \right. \\ &\left. + \frac{\partial \log P(y_i \mid X, \mathbf{y}_{-i}, \Theta^{(t)}, \eta, z_i = k)}{\partial \mathrm{cov}\left(f_i^k\right)} \frac{\partial \mathrm{cov}\left(f_i^k\right)}{\partial \theta_j^l} \right) \\ =& \sum_{i=1}^n w_i^l \frac{1}{[(K^l)^{-1}]_{ii}} \left(\alpha_i^l [Z_j^l \boldsymbol{\alpha}^l]_i \right. \\ &\left. - \frac{1}{2}\left(1 + \frac{(\alpha_i^l)^2}{[(K^l)^{-1}]_{ii}}\right)[Z_j^l (K^l)^{-1}]_{ii} \right),\end{aligned}$$

using the chain rule for derivation [20]. $\tilde{Q}$ is then optimized w.r.t. Θ using, for instance, scaled conjugate gradient optimization. We consider a constant noise fixed to the value learned by a standard GP on the entire regression dataset. After convergence of the proposed EM algorithm, all hyperparameters for the resulting GP experts are finally learned again including the noise variances σ_k^2.

Learning Final Hyperparameters: To refine the hyperparameters $\theta^1, ..., \theta^R$ and $\sigma_1^2, ...\sigma_R^2$ of the expert GPs $\mathcal{GP}_k$, we divide the regression dataset into expert training sets, one for each expert. The splits are based on a threshold t of the mixture weights which were learned by the EM algorithm. Then, a standard GP is trained on the corresponding set to learn the experts' final hyperparameters.

6 Experimental Illustration

Our intention here is to investigate the power of intertwining probabilistic logical and GP models, i.e., the following questions: **(Q1)** Can MLxGP improve performance upon (M)GPs? **(Q2)** Can MLxGP improve performance upon MLNs? **(Q3)** Can MLxGP be used



(a) Kitchen (b) Terrain (c) Friends-Coffee (d) Motorcycle

Figure 4: **Experimental Results** (a) shows segmentations for the Kitchen example indicating which object was manipulated at which time, MLN (top) and MLxGP (bottom). (b) compares the regression predictions of GPR (top) and MLxGP (bottom) for the Terrain example. In (c), the MLxGP (top) and the segmenting gating functions (bottom) are illustrated for the Friends-Coffee example. The regression predictions for the Motorcycle dataset is depicted in (d), MGP (top) and MLxGP (bottom).

to realize "machines reading regression data"?

To answer **(Q1)-(Q3)**, we conducted experiments on several datasets: the real-world Kitchen data used for illustration throughout the paper, synthetic datasets for a 1D terrain model and a friends-coffee model, and the motorcycle dataset. Specifically, we compare MLxGP and MGP **(Q1)** on a standard MGP benchmark dataset, namely the motorcycle dataset. To compare with GPs, we use leave-one-out cross-validation (LOO-CV) to compute normalized root mean squared error (*NRMSE*) and negative log predictive density (*NLPD*). For both measures holds, the smaller the better. To compare with MLNs, we compute the conditional marginal log-likelihood (*CMLL*) defined as $CMLL(X = x) = \sum_{i \in Q} \log P(X_i = x_i \mid E)$ with query set Q and evidence set E. *CMLL* is zero for a perfect prediction and negative otherwise. To demonstrate that enriching an MLN with continuous regression data will improve upon the belief estimate of certain query variables, we use fixed sets Q and E rather than performing cross-validation. In all experiments, we set the rescaling parameter $r = 6$, the threshold $t = 0.5$, and used junction tree for inference in the MLN.

TUM Kitchen Data (Q2,Q3): We evaluated MLxGP on our Kitchen example which is based on the TUM Kitchen dataset [26]. Of the demonstration indexed by (1-3), we considered the time frames where the subject set the table with a `plate`, a `knife`, and a `cup`, and viewed a subsample of the motion capture data of the height of the subject's right hand as depicted in Fig. 1. The full MLN is provided in Tab. 1. The MLxGP experts represented whether an object was carried from the cupboard (blue) or the kitchen counter (red). MLxGP not only provides the corresponding segmentation (cf. Fig. 3) — thus (Q3) can be answered affirmatively — but it tells us *which* object was manipulated through the predicate `segment(Object,Time)`. Fig. 4(a) illustrates that the MLN (top) did not give good results since it does not take regression data into account, and hence is undecided which object was carried after the plate. MLxGP (bottom), in contrast, infers that the subject first carried the plate, then the knife, followed by the cup. In fact, MLxGP achieves a *CMLL* of -1.07 whereas MLN obtains a *CMLL* of -57.81, cf. Tab. 3. This clearly shows that it is beneficial to incorporate regression data into MLNs and hence, (Q2) can be answered affirmatively. Moreover, this advantage does not considerably affect regression performance, cf. Tab. 2.

1D Terrain Model (Q1): Suppose that we want to model a terrain consisting of flat regions and a hill/wall from n observations (x_i, y_i) where x_i models depth and y_i observed height. Due to limited view, there are no observations taken directly behind the wall which results in high local uncertainty when modeling the terrain. However, if we have further witnessed a head behind the wall, a human could easily reason how thick this wall is and that it is very likely to be flat ground behind the wall, cf. top of Fig. 4(b). MLxGP incorpo-

rates this kind of reasoning into GPR such that these implicit observations can efficiently be processed for regression prediction. Let us further suppose the following rules: (a) the flat region has a smooth surface and the wall has a rough texture, (b) when observing a head at location x at a certain height, e.g. at height 6, it is likely that at this location, the person is standing on flat ground. In the KB, we model these constraints by probabilistic first-order logic and provide evidence about the consistency of the surface at some locations, as well as about the observation of a head directly behind the wall.

To compare regression performance of MLxGP and GPR, we evaluated the models by taking the unobserved points behind the wall into account such that LOO-CV considers them as additional test sets. The error measures of MLxGP are $NRMSE = 0.04$ and $NLPD = 0.22$ in comparison with $NRMSE = 0.10$ and $NLPD = 3.57$ in the case of GPR as provided in Tab. 2, which answers (Q1) clearly affirmatively. These results present the power of MLxGPs to approximate unknown data points for which we have background knowledge, i.e. we can perform constraint reasoning with GPs.

Friends-Coffee (Q2,Q3): In our synthetic Friends-Coffee example, the regression dataset consists of water consumption measurements taken in a coffee kitchen at a research institute, cf. Fig. 4(c). We modeled the data using two GP experts indicating whether a coffee drinker is at the office or not. In the MLN, we encoded that if two persons are friends, they probably either are both coffee drinkers or not. We assumed two persons `anna` and `bob` and provided evidence that they are friends, `bob` is a coffee drinker, and some evidence about which days they were at work. When analyzing the regression data, a human could easily reason that `anna` is not a coffee drinker, although she is friends with `bob`. Now, running inference within the MLN yielded a probability of 0.57 for `anna` being a coffee drinker. The MLxGP, however, additionally takes the regression data into account and predicted a probability of 0.0 for the same query. This improvement is also shown by the *CMLL*: -18.20 for MLN and -2.96 for MLxGP, cf. Tab. 3. This clearly gives an affirmative answer to (Q2).

It also provides an affirmative answer to (Q3) as shown by the segmenting gating function in Fig. 4(c) (bottom). We can actually read off: "At first, Anna who drinks *no coffee*, [2] was *alone* at the office. After a few days, her *friend* Bob came to work, consuming a lot of water from the coffee kitchen as he is a *coffee drinker*. Then, he *worked one day from home*, and *returned* to the office at the next day. The following days, Anna was again *alone* at the office whilst Bob might have *returned* for one day, probably *without drinking coffee* at that day."

Motorcycle (Q1): A common benchmark dataset for MGPs is the motorcycle dataset [24]. This real-world dataset consists of measurements of acceleration of the head of a motorcycle rider at an impact. The data is divided into three phases (riding, impact, hitting ground) and therefore, in both MGP and MLxGP, we modeled the data using three GP experts. For MLxGP, we augmented the dataset with soft evidence on the mixtures, i.e. on every mixture weight, we set a probability which reflects our initial belief about the corresponding data point belonging to the respective expert. We further integrated natural constraints such as "the impact phase cannot be followed by the riding phase", etc. Fig. 4(d) compares the performance of MLxGP (top) with the one of MGP (bottom). As the colors indicate, MLxGP was able to partition the input space into the expected three phases and to depict the corresponding modes. In contrast, the experts of the MGP have no specific meaning. Moreover, this additional capability is not sacrificing regression performance. The MGP model yields an *NRMSE* of 0.10 and an *NLPD* of 4.62, MLxGP had an *NRMSE* of 0.08 and an *NLPD* of 4.28. This clearly provides an affirmative answer to (Q1).

7 Conclusions

We presented a novel mixture of GPs model, called MLxGP, which mixes several Gaussian process experts using Markov logic (ML). MLxGP can be used for defining complex non-i.i.d. models for tasks requiring both Bayesian non-parametric regression and probabilistic logical inference. We introduced the model and inference algorithms for it, and illustrated its effectiveness on several datasets. Overall our contribution and results are an encouraging sign that "machines reading regression data" (MR^{2}D) may not be insurmountable.

This work suggests several avenues for future work such as relational networks of GPs lifting dependent GPs [2] to the relational level, learning the structure and the weights of the MLN jointly with the MGP part, and the application of MLxGPs within real-world applications, e.g. realizing robots that understand their grasps.

Acknowledgments: The authors thank the anonymous reviewers for their helpful comments. This work was partly supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

[2] We assume that only coffee drinkers consume water from the coffee kitchen frequently whereas others prefer to get water from somewhere else.

References

[1] M. Alvarez, J. Peters, B. Schölkopf, and N. Lawrence. Switched latent force models for movement segmentation. In *Proc. of Neural Information Processing Systems (NIPS-2010)*, pages 55–63. 2010.

[2] P. Boyle and M. Frean. Dependent Gaussian processes. In *Proc. of Neural Information Processing Systems (NIPS-2005)*, pages 217–224, 2005.

[3] W. Bridewell, P. Langley, L. Todorovski, and S. Dzeroski. Inductive process modeling. *Machine Learning*, 71(1):1–32, 2008.

[4] J. Choi, D. Hill, and E. Amir. Lifted inference for relational continuous models. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-2010)*, pages 126–134, 2010.

[5] W. Chu, V. Sindhwani, Z. Ghahramani, and S.S. Keerthi. Relational learning with Gaussian processes. In *Proc. of Neural Information Processing Systems (NIPS-2006)*, pages 289–296, 2006.

[6] L. De Raedt. *Logical and Relational Learning*. Springer, 2008.

[7] L. De Raedt, P. Frasconi, K. Kersting, and S.H. Muggleton, editors. *Probabilistic Inductive Logic Programming*, volume 4911 of *Lecture Notes in Computer Science*. Springer, 2008.

[8] M. P. Deisenroth, J. Peters, and C. E. Rasmussen. Approximate dynamic programming with Gaussian processes. In *Proc. of the American Control Conference (ACC-2008)*, pages 4480–4485, 2008.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

[10] O. Etzioni, M. Banko, and M. J. Cafarella. Machine reading. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI-2006)*, 2006.

[11] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[12] B. Gutmann, M. Jaeger, and L. De Raedt. Extending ProbLog with continuous distributions. In *Proc. of the 20th International Conference on Inductive Logic Programming (ILP-2010)*, pages 76–91, 2010.

[13] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[14] E. Meeds and S. Osindero. An alternative infinite mixture of Gaussian process experts. In *Proc. of Neural Information Processing Systems (NIPS-2006)*, pages 883–890, 2006.

[15] T.P. Minka and J.M. Winn. Gates. In *Proc. of Neural Information Processing Systems (NIPS-2008)*, pages 1073–1080, 2008.

[16] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proc. of the 15th Conference on Uncertainty in AI (UAI-1999)*, pages 467–475, 1999.

[17] M. Neumann, K. Kersting, Z. Xu, and D. Schulz. Stacked Gaussian process learning. In *Proc. of the 9th IEEE International Conference on Data Mining (ICDM-2009)*, pages 387–396, 2009.

[18] D. Poole. First-Order Probabilistic Inference. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 985–991, 2003.

[19] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Proc. of Neural Information Processing Systems (NIPS-2002)*, pages 881–888, 2002.

[20] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[21] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[22] J. Q. Shi, R. Murray-Smith, and D. M. Titterington. Hierarchical Gaussian process mixtures for regression. *Statistics and Computing*, 15(1): 31–41, 2005.

[23] R. Silva, W. Chu, and Z. Ghahramani. Hidden common cause relations in relational learning. In *Proc. of Neural Information Processing Systems (NIPS-2007)*, 2007.

[24] B. W. Silverman. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society B*, 47(1):1–52, 1985.

[25] C. Stachniss, C. Plagemann, and A. J. Lilienthal. Learning gas distribution models using sparse Gaussian process mixtures. *Autonomous Robots*, 26(2–3):187–202, 2009.

[26] M. Tenorth, J. Bandouch, and M. Beetz. The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *Proc. of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Worskhops), Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, pages 1089–1096, 2009.

[27] V. Tresp. Mixtures of Gaussian processes. In *Proc. of Neural Information Processing Systems (NIPS-2000)*, pages 654–660, 2000.

[28] J. Wang and P. Domingos. Hybrid markov logic networks. In *Proc. of the 23nd AAAI Conference on Artificial Intelligence (AAAI-2008)*, pages 1106–1111, 2008.

[29] O. Williams. A switched Gaussian process for estimating disparity and segmentation in binocular stereo. In *In Proc. of Neural Information Processing Systems (NIPS-2006)*, pages 1497–1504, 2006.

[30] Z. Xu, K. Kersting, and V. Tresp. Multi-relational learning with Gaussian processes. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-2009)*, pages 1309–1314, 2009.