Gaussian Logic for Predictive Classification

Ondřej Kuželka, Andrea Szabóová, Matěj Holec, and Filip Železný

Faculty of Electrical Engineering, Czech Technical University in Prague
Technicka 2, 16627 Prague, Czech Republic
{kuzelon2,szaboand,holecmat,zelezny}@fel.cvut.cz,

Abstract. We describe a statistical relational learning framework called Gaussian Logic capable to work efficiently with combinations of relational and numerical data. The framework assumes that, for a fixed relational structure, the numerical data can be modelled by a multivariate normal distribution. We demonstrate how the Gaussian Logic framework can be applied to predictive classification problems. In experiments, we first show an application of the framework for the prediction of DNA-binding propensity of proteins. Next, we show how the Gaussian Logic framework can be used to find motifs describing highly correlated gene groups in gene-expression data which are then used in a set-level-based classification method.

Keywords: Statistical Relational Learning, Proteomics, Gene Expression

1 Introduction

Modelling of relational domains which contain substantial part of information in the form of real valued variables is an important problem with applications in areas as different as bioinformatics or finance. So far there have not been many relational learning systems introduced in the literature that would be able to model multi-relational domains with numerical data efficiently. One of the frameworks able to work in such domains are hybrid Markov logic networks [21]. However, there is currently no known approach to learning structure of hybrid Markov logic networks which is mainly due to their excessive complexity. In this paper we describe a relatively simple framework for learning in rich relational domains containing numerical data. The framework relies on multivariate normal distribution for which many problems have tractable or even analytical solutions. Our novel system exploits regularities in covariance matrices (i.e. regularities regarding correlations) for construction of models capable to deal with variable number of numerical random variables. We mainly show how this novel system can be applied in predictive classification. We show that it can be applied to classification directly (Bayesian learning) or indirectly (feature construction for gene-expression data).

2 A Probabilistic Framework

In this section, we describe a probabilistic model which will constitute theoretical foundations for our framework. Let $n \in N$. If $\boldsymbol{v} \in R^n$ then v_i $(1 \leq i \leq n)$ denotes the i-th component of $\boldsymbol{v}$. If $I \subseteq [1;n]$ then $\boldsymbol{v}_I = (v_{i_1}, v_{i_2}, \ldots v_{i_{|I|}})$ where $i_j \in I$ $(1 \leq j \leq |I|)$. To describe training examples as well as learned models, we use a conventional first-order logic language $\mathcal{L}$ whose alphabet contains a distinguished set of constants $\{\mathsf{r}_1, \mathsf{r}_2, \ldots \mathsf{r}_n\}$ and variables $\{R_1, R_2, \ldots R_m\}$ $(n, m \in N)$. An *r-substitution* ϑ is any substitution as long as it maps variables (other than) R_i only to terms (other than) r_j. For the largest k such that $\{R_1/\mathsf{r}_{i_1}, R_2/\mathsf{r}_{i_2}, \ldots, R_k/\mathsf{r}_{i_k}\} \subseteq \vartheta$ we denote $I(\vartheta) = (i_1, i_2, \ldots i_k)$. A (Herbrand) *interpretation* is a set of ground atoms of $\mathcal{L}$. $I(H)$ $(I(\varphi))$ denotes the naturally ordered set of indexes of all constants r_i found in an interpretation H ($\mathcal{L}$-formula φ).

Our training examples have both *structure* and *real parameters*. An example may e.g. describe a measurement of the expression of several genes; here the structure would describe functional relations between the genes and the parameters would describe their measured expressions. The structure will be described by an interpretation, in which the constants r_i represent uninstantiated real parameters. The parameter values will be determined by a real vector. Formally, an example is a pair $(H, \boldsymbol{\theta})$ where H is an interpretation, $\boldsymbol{\theta} \in \Omega_H$, and $\Omega_H \subseteq R^{|I(H)|}$. The pair $(H, \boldsymbol{\theta})$ may also be viewed as a non-Herbrand interpretation of $\mathcal{L}$, which is the same as H except for including R in its domain and assigning θ_i to r_i.

Examples are assumed to be sampled from the distribution

$$P(H, \Omega_H) = \int_{\Omega_H} f_H(\boldsymbol{\theta}|H)\, P(H)\, \mathrm{d}\boldsymbol{\theta}$$

which we want to learn. Here, $P(H)$ is a discrete probability distribution on the countable set of finite Herbrand interpretations of $\mathcal{L}$. If $\mathcal{L}$ has functions other than constants, we assume that $P(H)$ is non-zero only for finite H. $f_H(\boldsymbol{\theta}|H)$ are the conditional densities of the parameter values. The advantage of this definition is that it cleanly splits the possible-world probability into the discrete part $P(H)$ which can be modeled by state-of-the-art approaches such as Markov Logic Networks (MLN's) [6], and the continuous conditional densities $f_H(\boldsymbol{\theta}|H)$ which we elaborate here. In particular, we assume that $f(\boldsymbol{\theta}|H) = N(\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H)$, i.e., $\boldsymbol{\theta}$ is normally distributed with mean vector $\boldsymbol{\mu}_H$ and covariance matrix $\boldsymbol{\Sigma}_H$. The indexes H emphasize the dependence of the two parameters on the particular Herbrand interpretation that is parameterized by $\boldsymbol{\theta}$.

To learn $P(H, \Omega_H)$ from a sample E, we first discuss a strategy that suggests itself readily. We could rely on existing methods (such as MLN's) to learn $P(H)$ from the multi-set $\mathcal{H}$ of interpretations H occurring in E. Then, to obtain $f(\boldsymbol{\theta}|H)$ for each $H \in \mathcal{H}$, we would estimate $\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H$ from the multi-set $\hat{\Omega}_H$ of parameter value vectors $\boldsymbol{\theta}$ associated with H in the training sample E. The problem of this approach is that, given a fixed size of the training sample, when $\mathcal{H}$ is large, the multi-sets $\hat{\Omega}_H, H \in \mathcal{H}$ will be small, and thus the estimates of $\boldsymbol{\mu}_H, \boldsymbol{\Sigma}_H$ will

be poor. For example, $\mathcal{H}$ may describe hundreds of metabolic pathway structures and each $\hat{\Omega}_H$ may contain a few vectors of expressions related to proteins acting in H, and measured through costly experiments. For this kind of problem we develop a solution here. We explore a method, in which parameters determining $P(H, \Omega_H)$ can be estimated using the entire training set. The type of $P(H, \Omega_H)$ is obviously not known; note that it is generally not a Gaussian mixture since the $\boldsymbol{\theta}$ in the normal densities $f_H(\boldsymbol{\theta}|H)$ have, in general, different dimensions for different H. However, our strategy is to learn *Gaussian features* of the training set. A Gaussian feature (*feature*, for short) is a $\mathcal{L}$-formula φ, which for each example $(H, \boldsymbol{\theta})$ extracts some components of $\boldsymbol{\theta}$ into a vector $\boldsymbol{u}(\varphi)$, such that $\boldsymbol{u}(\varphi)$ is approximately normally distributed across the training sample. For each feature φ, $\boldsymbol{\mu}_{\boldsymbol{u}(\varphi)}$ and $\boldsymbol{\Sigma}_{\boldsymbol{u}(\varphi)}$ are then estimated from the entire training sample. A set of such learned features φ can be thought of as a constraint-based model determining an approximation to $P(H, \Omega_H)$. We define *Gaussian features* more precisely in a moment after we introduce *sample sets*.

Given an example $e = (H, \boldsymbol{\theta})$ and a feature φ, the *sample set* of φ and e is the multi-set $\mathcal{S}(\varphi, e) = \{\boldsymbol{\theta}_{I(\vartheta)} | H \models \varphi\vartheta\}$ where ϑ are r-substitutions grounding all free variables[1] in φ, and $H \models \varphi\vartheta$ denotes that $\varphi\vartheta$ is true under H.

Now we can formally define *Gaussian features*. Let φ be a $\mathcal{L}$-formula, $\{e_i\}$ be a set of examples drawn independently from a given distribution and let $\boldsymbol{\theta_i}$ be vectors, each drawn randomly from $\mathcal{S}(\varphi, e_i)$. We say that φ is a *Gaussian feature* if $\boldsymbol{\theta_i}$ is multivariate-normally distributed[2].

Given a non-empty sample set $\mathcal{S}(\varphi, e)$, we define the *mean vector* as

$$\boldsymbol{\mu}(\varphi, e) = \frac{1}{|\mathcal{S}(\varphi, e)|} \sum_{\boldsymbol{\theta} \in \mathcal{S}(\varphi, e)} \boldsymbol{\theta} \tag{1}$$

and the $\boldsymbol{\Sigma}$-*matrix* as

$$\boldsymbol{\Sigma}(\varphi, e) = \frac{1}{|\mathcal{S}(\varphi, e)|} \sum_{\boldsymbol{\theta} \in \mathcal{S}(\varphi, e)} (\boldsymbol{\theta} - \boldsymbol{\mu}(\varphi, e)) (\boldsymbol{\theta} - \boldsymbol{\mu}(\varphi, e))^T \tag{2}$$

Finally, using the above, we define estimates over the entire training set $\{e_1, e_2, \ldots e_m\}$

$$\widehat{\boldsymbol{\mu}}_\varphi = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\mu}(\varphi, e_i) \tag{3}$$

$$\widehat{\boldsymbol{\Sigma}}_\varphi = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\Sigma}(\varphi, e_i) + \boldsymbol{\mu}(\varphi, e_i) \boldsymbol{\mu}(\varphi, e_i)^T \right) - \widehat{\boldsymbol{\mu}}_\varphi \widehat{\boldsymbol{\mu}}_\varphi^T \tag{4}$$

[1] Note that an interpretation H does not assign domain elements to variables in $\mathcal{L}$. The truth value of a *closed* formula (i.e., one where all variables are quantified) under H does not depend on variable assignment. For a general formula though, it does depend on the assignment to its free (unquantified) variables.

[2] Note that whether a $\mathcal{L}$-formula φ is a Gaussian feature depends also on the particular distribution of the examples.

Let us exemplify the concepts introduced so far using an example concerning modelling of gene regulatory networks. Let the only Herbrand interpretations H with non-zero $P(H)$ be those composed of literals of the form $g(G_i, R_i)$ and $expr(G_i, G_j)$ where $g(G_i, R_i)$ is a predicate intended to capture *expression level* R_i of a gene G_i and $expr(G_i, G_j)$ is used to indicate that two genes are in relation of *expression* (i.e. that the first gene G_i is a *transcription factor* of gene G_j). For example, the next example $e_1 = (H_1, \boldsymbol{\theta}_1)$ corresponds to a measurement on a sample set of genes containing three genes $H_1 = g(g_1, r_1)$, $g(g_2, r_2)$, $g(g_3, r_3)$, $expr(g_1, g_3)$, $expr(g_2, g_3)$, $\boldsymbol{\theta}_1 = (0, 1, 0)$. Let us further suppose that we have another example $e_2 = (H_2, \boldsymbol{\theta}_2)$ which corresponds to another set of genes: $H_2 = g(g_1, r_1)$, $g(g_2, r_2)$, $g(g_3, r_3)$, $g(g_4, r_4)$, $expr(g_1, g_2)$, $expr(g_2, g_3)$, $expr(g_3, g_4)$, $\boldsymbol{\theta}_2 = (1, 1, 0, 1)$.

Assume that the following two formulas have been identified as Gaussian features $\varphi_1 = g(G_1, R_1) \wedge g(G_2, R_2) \wedge expr(G_1, G_2)$, $\varphi_2 = g(G_1, R_1) \wedge g(G_2, R_2) \wedge \neg expr(G_1, G_2) \wedge G_1 \neq G_2$. Their sample sets for both examples are $\mathcal{S}(\varphi_1, e_1) = \{(0, 0), (1, 0)\}$, $\mathcal{S}(\varphi_2, e_1) = \{(0, 1), (1, 0)\}$, $\mathcal{S}(\varphi_1, e_2) = \{(1, 1), (1, 0), (0, 1)\}$ $\mathcal{S}(\varphi_2, e_2) = \{(1, 0), (0, 1), (1, 1), (1, 0), (1, 1), (1, 1), (0, 1), (1, 1), (1, 1)\}$. The first element of $\mathcal{S}(\varphi_1, e_1)$ is obtained with $\vartheta = \{G_1/g_1, G_2/g_3, R_1/r_1, R_2/r_3\}$. Clearly, $e_1 \models \varphi_1\vartheta$, and we have $\boldsymbol{\theta}_{I(\vartheta)} = (0, 1, 0)_{I(\vartheta)} = (0, 0)$ since $I(\vartheta) = (1, 3)$. For each sample set, we may then calculate the corresponding mean vector and $\boldsymbol{\Sigma}$-matrix according to Eq's 1 and 2 (e.g., $\boldsymbol{\mu}(\varphi_1, e_1) = (0.5, 0)^T$, and $\boldsymbol{\mu}(\varphi_1, e_2) = (2/3, 2/3)^T$). After that we can calculate the training-set-wide estimates for both features by Eq's 3 and 4. Then we can estimate multivariate normal distribution e.g. of a set of genes described by $H_3 = g(g_1, r_1)$, $g(g_2, r_2)$, $g(g_3, r_3)$, $expr(g_1, g_2)$, $expr(g_2, g_3)$, $expr(g_3, g_1)$ on the basis of features φ_1 and φ_2 which gives us the covariance matrix

$$\boldsymbol{\Sigma}_{H_3} = \begin{bmatrix} 1 & c_e & c_e \\ c_e & 1 & c_e \\ c_e & c_e & 1 \end{bmatrix}$$

where the parameter c_e corresponds to the correlation coefficient estimated in feature φ_1.

Importantly, using the training-set-wide estimates, we can derive estimates of parameters $\boldsymbol{\mu}_{H_n}$ and $\boldsymbol{\Sigma}_{H_n}$ of the densities $f_{H_n}(\boldsymbol{\theta}|H_n)$ for any relational structure H_n consisting of the two types of literals[3], even if H_n does not occur in the training set. Of course, validity of such estimates highly depends on the question whether the constructed features are truly *Gaussian*. Otherwise, a problem might occur that the estimated matrix would not be positive definite, however, first it almost did not happen in our experiments and second, if such a situation really occurs, one can replace the *defective* estimated covariance by a nearest positive definite matrix [10].

[3] Note that one can use any number of different predicate symbols in this framework, not just two.

3 Parameter Estimation

In this section, we will be concerned with estimation of parameters $\boldsymbol{\mu}_\varphi$ and $\boldsymbol{\Sigma}_\varphi$. Although the estimators are straightforward modifications of ordinary estimators of means and covariances, their correctness does not follow immediately from the correctness of these conventional estimators because the samples contained in sample sets $\mathcal{S}(\varphi, e_i)$ may be dependent. Namely, we will show that, for a Gaussian feature φ,

$$\widehat{\boldsymbol{\mu}}_\varphi = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{\mu}(\varphi, e_i)$$

is a consistent and unbiased estimator of the true mean $\boldsymbol{\mu}_\varphi$ and that

$$\widehat{\boldsymbol{\Sigma}}_\varphi = \frac{1}{m}\sum_{i=1}^{m} \left(\boldsymbol{\Sigma}(\varphi, e_i) + \boldsymbol{\mu}(\varphi, e_i)\boldsymbol{\mu}(\varphi, e_i)^T\right) - \widehat{\boldsymbol{\mu}}_\varphi \widehat{\boldsymbol{\mu}}_\varphi^T$$

is a consistent asymptotically unbiased estimator of the true covariance matrix $\boldsymbol{\Sigma}_\varphi$. In order to show this, we will need the next lemma.

Lemma 1. *Let $\mathcal{E}$ be a countable set of estimators converging in mean to the true value such that for each $\widehat{E}^i \in \mathcal{E}$, $\widehat{E}^j \in \mathcal{E}$ it holds $\mathbf{E}\widehat{E}^i_m = \mathbf{E}\widehat{E}^j_m$ (where $\widehat{E}^j_m$ denotes the estimate of the estimator $\widehat{E}^j$ using m-samples). Let $\mathcal{E}_i \subseteq \mathcal{E}$ be finite sets. Then*

$$\lim_{m\to\infty} Pr\left(|E^* - \frac{1}{|\mathcal{E}_m|}\sum_{\widehat{E}_i \in \mathcal{E}_m} \widehat{E}^i_m| > \epsilon\right) = 0$$

where $\epsilon > 0$ and E^* is the true value (i.e. the combination of the estimators is consistent).

First, we will rewrite the formula for $\widehat{\boldsymbol{\mu}}_\varphi$ as an average of a (large) number of consistent estimators converging in mean and after that we will apply Lemma 1. Let us impose a random total ordering on the elements of the sample sets $\mathcal{S}(\varphi, e_i) = \{s_1, \ldots, s_{m_i}\}$ so that we could index the elements of these sets. Next, let us have

$$\mathcal{X} = \{1, 2, \ldots, |\mathcal{S}(F, e_1)|\} \times \{1, 2, \ldots, |\mathcal{S}(\varphi, e_2)|\} \times \cdots \times \{1, 2, \ldots, |\mathcal{S}(\varphi, e_m)|\}$$

Then the formula for $\widehat{\boldsymbol{\mu}}_\varphi$ can be rewritten as follows:

$$\widehat{\boldsymbol{\mu}}_\varphi = \frac{1}{|\mathcal{X}|}\sum_{(i_1, i_2, \ldots, i_m)\in\mathcal{X}} \frac{1}{m}\left(s_{1,i_1} + s_{2,i_2} + \cdots + s_{m,i_m}\right)$$

where $s_{j,k}$ is a k-th element of the sample set $\mathcal{S}(\varphi, e_j)$. Now, each $\frac{1}{m}\left(s_{1,i_1} + s_{2,i_2} + \cdots + s_{m,i_m}\right)$ is an unbiased consistent estimator converging in mean according to the definition of Gaussian features (it is the ordinary estimator of mean). Now, we may apply Lemma 1 and infer that $\widehat{\boldsymbol{\mu}}_\varphi$ also converges in probability

to $\boldsymbol{\mu}_\varphi$. The unbiasedness of the estimator then follows from basic properties of expectation.

The argument demonstrating consistency and asymptotic unbiasedness of the covariance estimator goes along similar lines as the argument for the mean estimator. First, we rewrite the sum

$$\widehat{\boldsymbol{\Sigma}}_\varphi = \frac{1}{m}\sum_{i=1}^{m}\left(\boldsymbol{\Sigma}(\varphi, e_i) + \boldsymbol{\mu}(\varphi, e_i)\boldsymbol{\mu}(\varphi, e_i)^T\right) - \widehat{\boldsymbol{\mu}}_\varphi\widehat{\boldsymbol{\mu}}_\varphi^T =$$

$$= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathcal{S}(\varphi, e_i)|}\sum_{\boldsymbol{\theta_j}\in\mathcal{S}(\varphi,e_i)}\left(\boldsymbol{\theta_j} - \widehat{\boldsymbol{\mu}}_\varphi\right)\left(\boldsymbol{\theta_j} - \widehat{\boldsymbol{\mu}}_\varphi\right)^T$$

as a sum of asymptotically unbiased consistent estimators as follows:

$$\widehat{\boldsymbol{\Sigma}}_\varphi = \frac{1}{|\mathcal{X}|}\sum_{(i_1,\ldots,i_m)\in\mathcal{X}}\frac{1}{m}\Big(\left(s_{1,i_1} - \widehat{\boldsymbol{\mu}}\right)\left(s_{1,i_1} - \widehat{\boldsymbol{\mu}}\right)^T + \ldots$$

$$\ldots + \left(s_{n,i_m} - \widehat{\boldsymbol{\mu}}_\varphi\right)\left(s_{m,i_m} - \widehat{\boldsymbol{\mu}}_\varphi\right)^T\Big)$$

where $s_{j,k}$ is a k-th element of the sample set $\mathcal{S}(\varphi, e_j)$. Now, each sum

$$\frac{1}{m}\Big(\left(s_{1,i_1} - \widehat{\boldsymbol{\mu}}\right)\left(s_{1,i_1} - \widehat{\boldsymbol{\mu}}\right)^T + \cdots + \left(s_{m,i_m} - \widehat{\boldsymbol{\mu}}\right)\left(s_{m,i_m} - \widehat{\boldsymbol{\mu}}_\varphi\right)^T\Big)$$

is an asymptotically unbiased and consistent estimator converging in mean and the average of these estimators is thus asymptotically unbiased and consistent (again by basic properties of expectation and by Lemma 1).

In general, the problem of estimating $\boldsymbol{\mu}(\varphi, e_i)$ and $\boldsymbol{\Sigma}(\varphi, e_i)$ are NP-hard problems (they subsume the well-known NP-complete problem of θ-subsumption). However, they are tractable for a class of features, *conjunctive tree-like features* for which we have devised efficient algorithms (see Appendix for details).

4 Structure Search

In this section we briefly describe methods for constructing a set of features that *give rise* to models capable to appropriately model a given set of examples. The methods that we describe are specialized for working with tree-like features because estimation of the parameters is tractable for them as we have mentioned in the previous section. The feature construction algorithm for tree-like features is based on the feature-construction algorithm from [16]. It shares most of the favourable properties of the original algorithm like detection of redundant features. The output of the feature construction algorithm is a (possibly quite large) set of features and their parameters so we need to select a subset of these features which would provide us with good models.

First, let us describe how a mean vector and a covariance matrix for a relational structure (i.e. a Herbrand interpretation) R is computed using a given

set of features. We assume that we have a set of features $\varphi_i \in \mathcal{F}$ which have been identified as Gaussian, their respective parameters $\boldsymbol{\mu}_{\varphi_i}$ and $\boldsymbol{\Sigma}_{\varphi_i}$ and a relational structure H (Herbrand interpretation) for which we want to construct the model. Furthermore, we assume that each distinguished constant r_i contained in H *is covered* by some feature $\varphi \in \mathcal{F}$, i.e. that for each r_i there is a feature $\varphi \in \mathcal{F}$ and a r-substitution ϑ such that r_i is contained in $\varphi\vartheta$. Then the covariance matrix $\boldsymbol{\Sigma}_H$ can be constructed as follows. For each $\varphi \in \mathcal{F}$ we compute the set Θ_φ of all substitutions ϑ such that $H \models \varphi\vartheta$. After that, for each feature φ (with parameters $\boldsymbol{\mu}_\varphi$ and $\boldsymbol{\Sigma}_\varphi$) and each r-substitution $\vartheta \in \Theta_\varphi$, we set the entries $(\boldsymbol{\mu}_H)_i = (\boldsymbol{\mu}_\varphi)_I$ $(\boldsymbol{\Sigma}_H)_{i,j} = (\boldsymbol{\Sigma}_\varphi)_{I,J}$ where $\{R_I/r_i, R_J/r_j\} \subseteq \vartheta$. If the features are perfectly Gaussian and if the parameters $\boldsymbol{\mu}_\varphi$ and $\boldsymbol{\Sigma}_\varphi$ are known accurately, there is no problem. However, in practice, we may encounter situations where two features will *suggest* different values for some entries (be it for the reason that the features are not perfectly Gaussian or that the parameters were estimated from small samples). In such situations we will use an average of the values *suggested* by different features.

Having explained how $\boldsymbol{\mu}_H$ and $\boldsymbol{\Sigma}_H$ are constructed using a set of Gaussian features, we can explain a simple procedure for construction of the Gaussian feature set. On the input, we get a set of examples $e_i = (H_i, \boldsymbol{\theta}_i)$. The procedure starts by constructing a large set of *non-redundant* features exhaustively on a subset of the training data. Then, in the second step, a subset of features is selected. This is done by a greedy search algorithm optimizing a score function of the models on a different subset of training data not used previously for feature construction and parameter estimation.

5 A Straightforward Predictive Classification Method

A straightforward application of the Gaussian-logic framework is Bayesian classification. We use the algorithms described in the previous sections of this paper to learn a Gaussian-logic model for positive examples and a Gaussian-logic model for negative examples and then we use these models to classify examples by comparing likelihood ratios of the two models with a threshold. In this section we describe a case study involving an important problem from biology - prediction of DNA-binding propensity of proteins. Proteins which possess the ability to bind to DNA play a vital role in the biological processing of genetic information like DNA transcription, replication, maintenance and the regulation of gene expression. Several computational approaches have been proposed for the prediction of DNA-binding function from protein structure. It has been shown that electrostatic properties of proteins such as total charge, dipole moment and quadrupole moment or properties of charged patches located on proteins' surfaces are good features for predictive classification (e.g. [1], [3], [18], [20]). Szilágyi and Skolnick [19] created a logistic regression classifier based on 10 features including electrostatic dipole moment, proportions of charged amino acids Arg, Lys and Asp, spatial asymmetries of Arg and five more features not related to charged

amino-acids: proportion of Ala and Gly and spatial asymmetry of Gly, Asn and Ser.

Here, we use Gaussian logic to create a model for capturing distributions of positively charged amino acids in protein sequences. Clearly, the distinguishing electrostatic properties of DNA-binding proteins, which have been observed in 3D structures of proteins in the previous works, should exhibit themselves also in the amino-acid protein sequences (possibly, not in a straightforward manner because the 3D structure is a result of complicated folding of a protein's sequence). We split each protein into consecutive non-overlapping *windows*, each containing l_w amino acids (possibly except for the last window which may contain less amino acids). For each window of a protein P we compute the value a_i^+/l_w where a_i^+ is the number of positively charged amino-acids in the window i. Then for each protein P we construct an example $e_P = (H_P, \boldsymbol{\theta}_P)$ where $\boldsymbol{\theta}_P = \left(a_1^+/l_w, a_2^+/l_w, \ldots, a_{n_P}^+/l_w\right)$ and $H_P = w(1, r_1)$, $next(1, 2)$, $\ldots$, $next(n_P - 1, n_P)$, $w(n_P, r_P)$. We constructed only one feature $F_{non} = w(A, R_1)$ for non-DNA-binding proteins since we do not expect this class of proteins to be very homogeneous. For DNA-binding proteins, we constructed a more complex model by selecting a set of features using a greedy search algorithm. The greedy search algorithm optimized classification error on training data. Classification was performed by comparing, for a tested protein, the likelihood-ratio of the two models (DNA-binding and non-DNA-binding) with a threshold selected on the training data. We estimated the accuracy of this method using 10-fold cross-validation (always learning parameters and structure of the models and selecting the threshold and window length l_w using only the data from training folds) on a dataset containing 138 DNA-binding proteins (PD138 [19]) and 110 non-DNA-binding proteins (NB110 [1]). The estimated accuracies (*Gaussian Logic*) are shown in Table 1. The method performs similarly well as the method of Szilagyi et al. [19] (in fact, it outperforms it slightly but the difference is rather negligible) but uses much less information. Next, we were interested in the question whether the machinery of Gaussian logic actually helped improve the predictive accuracy in our experiments or whether we could obtain the same or better results using only the very simple feature $F = w(A, R_1)$ also to model the DNA-binding proteins, thus ignoring any correlation between charges of different parts of a protein (*Baseline Gaussian Logic* in Table 1). Indeed, the machinery of Gaussian Logic appears to be helpful from these results.

Method	**Accuracy [%]**
Szilágyi et al.	81.4
Baseline Gaussian logic	78.7
Gaussian logic	**81.9**

Table 1. Accuracies estimated by 10-fold cross-validation on PD138/NB110.

It is interesting how well the Gaussian-logic model performed considering the fact that it used so little information (it completely ignored types of positively charged amino acids and it also ignored negative amino acids). The model that we presented here can be easily extended, e.g. by adding secondary-structure information. The splitting into consecutive windows used here is rather artificial and it would be more natural to split the sequence into windows corresponding to secondary-structure units (helices, sheets, coils). The features could then distinguish between consecutive windows corresponding to different secondary-structure units.

6 Feature Construction for Predictive Classification

In this section we present another application of the Gaussian-logic framework for predictive classification. We show how to use it to search for novel definitions of gene sets with high discriminative ability. This is useful in set-level classification methods for prediction from gene-expression data [11]. Set-level methods are based on aggregating values of gene expressions contained in pre-defined gene sets and then using these aggregated values as features for classification. Here, we, first, describe the problem and available data and then we explain how we can construct meaningful novel gene sets using Gaussian Logic.

The datasets contain class-labeled gene-samples corresponding to measurements of activities of thousands of genes. Typically, the datasets contain only tens of measured samples. In addition to this *raw* measured data, we also have relational description of some biological pathways from publicly available database KEGG [14]. Each KEGG pathway is a description of some biological process (a metabolic reaction, a signalling process etc.). It contains a set of genes annotated by relational description which contains relations among genes such as *compound*, *phosphorylation*, *activation*, *expression*, *repression* etc. The relations do not necessarily refer to the processes involving the genes per se but they may refer to relations among the products of these genes. For example, the relation *phosphorylation* between two genes A, B is used to indicate that a protein coded by the gene A adds phosphate group(s) to a protein coded by the gene B.

We constructed examples $(H_S, \boldsymbol{\theta}_S)$ from the gene-expression samples and KEGG pathways as follows. For each gene g_i, we introduced a logical atom $g(g_i, r_i)$ to capture its expression level. Then we added all relations extracted from KEGG as logical atoms $relation(g_i,\ g_j,\ relationType)$. We also added a numerical indicator of class-label to each example as a logical atom $label(\pm 1)$ where $+1$ indicates a positive example and -1 a negative example. Finally, for each gene-expression sample S we constructed the vector of the gene-expression levels $\boldsymbol{\theta}_S$. Using the feature construction algorithm outlined in Section 4 we constructed a large set of tree-like features[4] involving exactly one atom $label(L)$, at least one atom $g(G_i, R_i)$ and relations *expression*, *repression*, *activation*, *inhibition*, *phosphorylation*, *dephosphorylation*, *state* and *binding/association*. After

[4] We have used a subset of 50 pathways from KEGG to keep the memory consumption of the feature-construction algorithm under 1GB.

that we had to select a subset of these features. Clearly, the aggregated values of meaningful gene sets should correlate with the class-label. A very often used aggregation method in set-level classification methods is the *average*. Therefore what we need to do is to select features based on the correlation of the average expression of the genes assumed by the feature and the class-label but this is easy since we have the estimate of the features' covariance matrices $\mathbf{\Sigma}_\varphi$ and computing the average expression of the assumed genes is just an affine transform. It suffices to extract correlation from the covariance matrix given as $B\mathbf{\Sigma}_\varphi B^T$ where B is a matrix representing the *averaging*. The absolute values of correlations give us means to heuristically order the features. Based on this ordering we found a collection of gene sets given by the features (ignoring gene sets which contained only genes contained in a union of already constructed gene sets).

Dataset	**Gaussian logic**	**FCF**
Collitis [4]	80.0	89.4
Pleural Mesothieloma [9]	94.4	92.6
Parkinson 1 [17]	52.7	54.5
Parkinson 2 [17]	66.7	63.9
Parkinson 3 [17]	62.7	77.1
Pheochromocytoma [5]	64.0	56.0
Prostate cancer [2]	85.0	80.0
Squamus cell carcinoma [15]	95.5	88.6
Testicular seminoma [8]	58.3	61.1
Wins	5	4

Table 2. Accuracies of set-level-based classifiers with Gaussian-logic features and FCF-based features, estimated by leave-one-out cross-validation.

We have constructed the features using a gene-expression dataset from [7] which we did not use in the subsequent predictive classification experiments. A feature defining gene sets which exhibited one of the strongest correlations with the class-label was the following:

$$F = \mathit{label}(R_1) \wedge g(A, R_2) \wedge \mathit{relation}(A, B, \mathit{phosphorylation}) \wedge \\ g(B, R_3) \wedge \mathit{relation}(A, C, \mathit{phosphorylation}) \wedge g(C, R_4)$$

We have compared gene sets constructed by the outlined procedure with gene sets based on so called *fully-coupled fluxes (FCFs)* which are biologically-motivated gene sets used previously in the context of set-level classification [11]. We constructed the same number of gene sets for our features as was the number of FCFs. The accuracies of an SVM classifier (estimated by leave-one-out cross-validation) are shown in Table 2. We can notice that the gene sets constructed using our novel method performed equally well as the gene sets based on fully-coupled fluxes. Interestingly, our gene sets contained about half the number of

genes as compared to FCFs and despite that they were able to perform equally well.

7 Conclusions and Future Work

In this paper we have introduced a novel relational learning system capable to work efficiently with combinations of relational and numerical data. The experiments with real-world gene-expression and proteomics data gave us some very promising results. Furthermore, there are other possible applications of Gaussian logic in predictive classification settings which were not discussed in this paper. For example, finding patterns that generally correspond to highly correlated sets (not necessarily correlated with the class) of genes may have applications with group-lasso based classification approaches [12].

Acknowledgement: We thank the anonymous reviewers for their very valuable comments. This work was supported by the Czech Grant Agency through project 103/10/1875 *Learning from Theories* and project 103/11/2170 *Transferring ILP techniques to SRL*.

Appendix

In this appendix, we describe technical details concerning estimation of μ-vectors and $\mathbf{\Sigma}$-matrices.

Proof of Lemma 1

Let us suppose, for contradiction, that the assumptions of the lemma are satisfied, $\delta > 0$ and that

$$\delta = \lim_{n\to\infty} Pr\left(|E^* - \frac{1}{|\mathcal{E}_n|}\sum_{\widehat{E}_i\in\mathcal{E}_n}\widehat{E}_n^i| > \epsilon\right) \leq \lim_{n\to\infty} Pr\left(\frac{1}{|\mathcal{E}_n|}\sum_{\widehat{E}_i\in\mathcal{E}_n}|E^* - \widehat{E}_n^i| > \epsilon\right)$$

From this we have

$$\lim_{n\to\infty} \mathbf{E}\left(\frac{1}{|\mathcal{E}_n|}\sum_{\widehat{E}_i\in\mathcal{E}_n}|E^* - \widehat{E}_n^i|\right) \geq \delta\cdot\epsilon > 0$$

but

$$\lim_{n\to\infty} \mathbf{E}\left(\frac{1}{|\mathcal{E}_n|}\sum_{\widehat{E}_i\in\mathcal{E}_n}|E^* - \widehat{E}_n^i|\right) = \lim_{n\to\infty}\left(\frac{1}{|\mathcal{E}_n|}\sum_{\widehat{E}_i\in\mathcal{E}_n}\mathbf{E}|E^* - \widehat{E}_n^i|\right) =$$
$$= \lim_{n\to\infty}\left(\frac{1}{|\mathcal{E}_n|}\cdot|\mathcal{E}_n|\cdot\mathbf{E}|E^* - \widehat{E}_n^i|\right) = \lim_{n\to\infty}\left(\mathbf{E}|E^* - \widehat{E}_n^i|\right) = 0$$

(where the last equality results from the convergence in mean of the individual estimators) which is a contradiction. The only remaining possibility would be that the limit does not exist but then we can select a subsequence of $\mathcal{E}_i$ which has a non-zero limit and again derive the contradiction as before. □

Parameter Estimation

In this section we describe an efficient algorithm for estimation of μ-vectors and $\mathbf{\Sigma}$-matrices (polynomial in the combined size of a feature and an example) for the class of *tree-like conjunctive features*. Algorithms for computing quantities related to $\boldsymbol{\mu}(\varphi, e_i)$ for *tree-like features* have already been described in literature on relational aggregation [13]. However, there has been no prior work concerned with tractable computation of $\mathbf{\Sigma}(\varphi, e_i)$ or any similar quantity.

Definition 1 (Tree-like conjunction). *A first-order conjunction without quantifications C is* tree-like *if the iteration of the following rules on C produces the empty conjunction: (i) Remove an atom which contains fewer than 2 variables. (ii) Remove a variable which is contained in at most one atom.*

Intuitively, a tree-like conjunction can be imagined as a tree with the exception that whereas trees are graphs, conjunctions correspond in general to hypergraphs.

We start by some auxiliary definitions. Let φ be a tree-like feature. Let us suppose that $s_1, s_2, \ldots, s_k$ is a sequence of steps of the reduction procedure from Definition 1 which produces an empty feature from φ. Let $\lhd$ be an order on the atoms of φ such that if an atom a_1 disappeared before an atom a_2 during the reduction process then $a_1 \lhd a_2$. Then we say that $\lhd$ is a topological ordering of φ's atoms. Let $A \subseteq \varphi$ be a maximal set of atoms having a variable v in common. We say that $a \in A$ is a parent of atoms from $A \setminus \{a\}$ if for each $x \in A \setminus \{a\}$ it holds $x \lhd a$ (we also say that the atoms in $A \setminus \{a\}$ are a's children). An atom a is called root if it has no parents w.r.t. $\lhd$, it is called a leaf if it has no children w.r.t. $\lhd$.

We will use $(C, v) \in Children(\varphi, \lhd)$ for the set of all features φ_C with roots equal to children of φ (w.r.t. $\lhd$) together with the respective shared variables v. Similarly, we will use $C \in Children(\varphi, v, \lhd)$ for the set of all features φ_C with roots equal to children of φ (w.r.t. $\lhd$) sharing the variable v with φ's root. We will also use $arg_i(a)$ to identify the term appearing in the i-th argument of a. Next, to denote the set of all arguments of $a = atom(a_1, a_2, \ldots, a_k)$ and their positions in the atom we will use $(a_i, i) \in args(a)$. Finally, we define the *input* variable of a feature φ contained in some bigger feature ψ (denoted by $inp(\varphi, \psi, \lhd)$) as the variable which is shared by $root(\varphi, \lhd)$ with its parent in ψ. When a is a ground atom such that $root(\varphi, \lhd)\theta = a$ then we define *input* operator $inp(a, \varphi, \psi, \lhd)$ which will give us $inp(\varphi, \psi, \lhd)\theta$ (i.e. the term from the argument corresponding to the input variable in φ). If $\varphi = \psi$ then $inp(a, \varphi, \psi, \lhd) = inp(\varphi, \psi, \lhd) = \emptyset$[5].

[5] Here, the empty set is used as a *dummy* input. It does not mean that the returned value of the *input* functions would be a set in general.

We say that ψ is a sub-feature (of φ) if $\psi \subseteq \varphi$ and ψ and $\varphi \backslash \psi$ are both connected features. The parameter estimation algorithm will use two auxiliary algorithms for computing so-called *domains* and *term-domains* of features which are defined as follows.

Definition 2 (Domain, term-domain). *Let e be an example. Let φ be a tree-like feature, $\lhd$ be a topological ordering of φ's atoms. Then we say that a set of atoms $A \subseteq e$ is domain of φ w.r.t. e (denoted by $A = \mathcal{D}(\varphi, e, \lhd)$) if A contains all ground atoms $a = root(\varphi, \lhd)\theta$ such that $e \models \varphi\theta$. If φ is contained in a bigger feature ψ then we can also define its term-domain as $\mathcal{D}_T(\varphi, \psi, e, \lhd) = \{inp(\varphi, \psi, \lhd)\theta | e \models \varphi\theta\}$.*

Let $e = a(a,b), a(a,c), b(b)$ be an example and let $\varphi = a(X,Y), b(Y)$ and $\psi = a(X,Y), a(X,Z), b(Y)$ be features. Let $b(Y) \lhd a(X,Y)$. Then $\mathcal{D}(\varphi, e, \lhd) = \{a(a,b)\}$ and $\mathcal{D}_T(\varphi, \psi, e, \lhd) = \{a\}$.

Algorithms for computing domains and term-domains have been described in [16] and they also correspond to well-known algorithms for answering acyclic conjunctive queries [22]. These algorithms run in time polynomial in $|\varphi|$ and $|e|$. We note that these algorithms for computing domains of tree-like features compute not only domains corresponding to the roots of the given features but also domains corresponding to all the sub-features during one pass over a given feature. In the pseudocode of the parameter-estimation algorithms we will *call* the procedure for computing domains using $\mathcal{D}(\varphi, e, \lhd, T)$ where φ is the feature and e is the example for which we want to compute the domain, $\lhd$ is a topological ordering of φ's atoms and T is a table in which domains of all φ's sub-features should be stored. Similarly, we will *call* the procedure for computing term-domains using $\mathcal{D}_T(\varphi, \psi, e, \lhd, T)$ where again φ and e are the feature and the example for which we want to compute the term-domain, ψ is a feature containing φ (recall the definition of term-domain), $\lhd$ is a topological ordering of ψ's atoms and T is a table in which the computed term-domains of φ's sub-features should be stored.

Now, we can proceed further to computation of the parameters $\boldsymbol{\mu}(\varphi, e)$ and $\boldsymbol{\Sigma}(\varphi, e)$. By *sample parameters* we will mean a 5-tuple $(x, \widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}, n, \gamma)$. Here x can be either an empty set, an atom or a term, $\boldsymbol{\mu}$ is a vector, $\boldsymbol{\Sigma}$ is a matrix and n is a natural number. The parameter γ is an ordered list of the distinguished variables $R_i \in vars(\varphi)$. Next, we define a concatenation operation for combining sample parameters.

Definition 3 (Concatenation operator). *Let $A = (x, \boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A, n_A, \gamma_A)$ and $B = (y, \boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B, n_B, \gamma_B)$ be sample parameters. Then we define $A \otimes B$ as*

$$A \otimes B = \left(x, \left[\, \boldsymbol{\mu}_A^T \; \boldsymbol{\mu}_B^T \,\right]^T, \boldsymbol{\Sigma}_{AB}, n_A \cdot n_B, \gamma_A \cup \gamma_B\right)$$

where $\gamma_A \cup \gamma_B$ denotes concatenation of the lists γ_A and γ_B and $\boldsymbol{\Sigma}_{AB}$ is a block-diagonal matrix

$$\boldsymbol{\Sigma}_{AB} = \begin{bmatrix} \boldsymbol{\Sigma}_A & 0 \\ 0 & \boldsymbol{\Sigma}_B \end{bmatrix}$$

Algorithm 1 An algorithm for computing $\boldsymbol{\mu}(\varphi, e)$ and $\boldsymbol{\Sigma}(\varphi, e)$ for connected φ

Procedure: $\mu\sigma(\varphi, e = (H, \boldsymbol{\theta}), \lhd)$
1: $T \leftarrow []$
2: $\mathcal{D}(\varphi, e, \lhd, T)$ /* *This fills values into the table* T */
3: **return** $\bigoplus_{\lhd}^{\varphi,\varphi} \mu\sigma'(\varphi, \varphi, e, \lhd, T)$

Procedure: $\mu\sigma'(\varphi, \psi, e = (H, \boldsymbol{\theta}), \lhd, T)$
1: $SP \leftarrow []$ /* SP *is an associative array of sample parameters* */
2: $D_\varphi \leftarrow T[\varphi]$ /* D_φ *is domain of* φ */
3: **for** $\forall a \in D_\varphi$ **do**
4: $\quad SP[a] \leftarrow \{a, \boldsymbol{\theta}_{I(a)}, |I(a)|\times|I(a)| \text{ zero matrix}, 1, \mathcal{R}_a\}$ /* where $\mathcal{R}_a$ is a list of the distinguished variables contained in $root(\varphi)$ */
5: **end for**
6: **for** $(\varphi_C, v) \in Children(\varphi, \lhd)$ **do**
7: $\quad SP_{\varphi_C} \leftarrow \bigoplus_{\lhd}^{\varphi_C,\varphi} \mu\sigma'(\varphi_C, \varphi, e, \lhd)$
8: $\quad$ **for** $\forall a \in D_\varphi$ **do**
9: $\quad\quad SP[a] \leftarrow SP[a] \bigotimes SP_{\varphi_C}[v\vartheta]$ where $root(\varphi, \lhd)\vartheta = a$
10: $\quad$ **end for**
11: **end for**
12: **return** SP

Definition 4 (Combination operator). *Let $\varphi \subseteq \psi$ be features and $\lhd$ a topological ordering of ψ's atoms. Let γ be a list of distinguished variables $R_i \in vars(\varphi)$. Let $A = (x, \boldsymbol{\mu}_A, \boldsymbol{\Sigma}_A, n_A, \gamma)$ and $B = (y, \boldsymbol{\mu}_B, \boldsymbol{\Sigma}_B, n_B, \gamma)$ be sample parameters where x and y are logic atoms such that $inp(x, \varphi, \psi, \lhd) = inp(y, \varphi, \psi, \lhd)$. Then we define $A \oplus_{\lhd}^{\varphi,\psi} B$ as $A \oplus_{\lhd}^{\varphi,\psi} B = (inp(x, \varphi, \psi, \lhd), \boldsymbol{\mu}_{AB}, \boldsymbol{\Sigma}_{AB}, n_A + n_B, \gamma)$ where $\boldsymbol{\mu}_{AB} = \frac{1}{n_A+n_B}(n_A \cdot \boldsymbol{\mu}_A + n_B \cdot \boldsymbol{\mu}_B)$ and*

$$\boldsymbol{\Sigma}_{A,B} = \frac{1}{n_A + n_B}\left(n_A \cdot (\boldsymbol{\Sigma}_A + \boldsymbol{\mu}_A \cdot \boldsymbol{\mu}_A^T) + n_B \cdot (\boldsymbol{\Sigma}_B + \boldsymbol{\mu}_B \cdot \boldsymbol{\mu}_B^T)\right) - \boldsymbol{\mu}_{AB} \cdot \boldsymbol{\mu}_{AB}^T$$

Let us note that $\oplus_{\lhd}^{\varphi,\psi}$ is a commutative and associative operation which is also implicitly used in the following definition.

Definition 5 (Combination operator). *Let $\varphi \subseteq \psi$ be features and $\lhd$ a topological ordering of φ's atoms. Next, let γ be a list of distinguished variables $R_i \in vars(\varphi)$. Let $\mathcal{X} = \{(x_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, n_1, \gamma), \ldots, (x_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, n_k, \gamma)\}$. Next, let $\mathcal{X}[t]$ denote the set of all sample parameters $(x, \ldots) \in \mathcal{X}$ for which $inp(x, \varphi, \psi, \lhd) = t$. Then $\bigoplus_{\lhd}^{\varphi,\psi} \mathcal{X}$ is defined as follows:*

$$\bigoplus_{\lhd}^{\varphi,\psi} \mathcal{X} = \{(y_1, \boldsymbol{\mu}_{y_1}, \boldsymbol{\Sigma}_{y_1}, n_{y_1}, \gamma), \ldots, (y_m, \boldsymbol{\mu}_{y_m}, \boldsymbol{\Sigma}_{y_m}, n_{y_m}, \gamma)\}$$

where $(y_i, \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}, n_{y_i}, \gamma) = x_1 \oplus_{\lhd}^{\varphi,\psi} x_2 \oplus_{\lhd}^{\varphi,\psi} \cdots \oplus_{\lhd}^{\varphi,\psi} x_o$ for $\{x_1, \ldots, x_o\} = \mathcal{X}[y_i]$.

The basic ideas underlying Algorithm 1 are summarized by the next two observations.

Observation 1 *Let $\varphi = \psi \cup C_1 \cup \cdots \cup C_n$ be a feature where each C_i is a subfeature of φ and $\psi \cap C_i = \emptyset$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. Let ϑ be a substitution affecting only variables $v \in vars(\psi)$ and guaranteeing that $\psi\vartheta$ will be ground and*

it will hold $e \models \varphi\vartheta$ *where* $e = (H, \boldsymbol{\theta})$ *is an example. Then* $\boldsymbol{\mu}(\varphi\vartheta, e)$ *and* $\boldsymbol{\Sigma}(\varphi\vartheta, e)$ *and the number of samples* $m = |\mathcal{S}(\varphi\vartheta, e)|$ *are given by the sample parameters* A *(up to reordering of random variables) computed as*

$$A = (\psi\vartheta, \boldsymbol{\theta}_I(\vartheta), n \times n \text{ zero matrix}, 1, \mathcal{R}_\psi) \otimes \ldots$$
$$\otimes \mu\sigma(C_1\vartheta, e, \lhd\vartheta) \otimes \mu\sigma(C_2\vartheta, e, \lhd\vartheta) \otimes \cdots \otimes \mu\sigma(C_k\vartheta, e, \lhd\vartheta)$$

(where $n = |I(\psi)|$ *and* $\mathcal{R}_\psi$ *is a list of the distinguished variables* $R_i \in vars(\psi)$*) provided that* $\mu\sigma(C_i\vartheta, e, \lhd\vartheta)$ *are correct sample parameters.*

Let us look more closely at what $\mu\sigma(C_i\vartheta, e, \lhd\vartheta)$ is. First, we can notice that $C_i\vartheta$ is a sub-feature of φ which differs from C_i only by the fact that it has its input variable ($inp(C_i, \varphi, \lhd)$) grounded by ϑ. Therefore $\mu\sigma(C_i\vartheta, e, \lhd\vartheta)$ can be also obtained from the set $\bigoplus_{\lhd}^{C_i,\varphi} \mu\sigma'(C_i, \varphi, e, \lhd, T)$ (where the argument T is a table containing pre-computed domains). The next observation, in turn, shows that what $\bigoplus_{\lhd}^{C_i,\varphi} \mu\sigma'(C_i, \varphi, e, \lhd, T)$ contains are the sample parameters corresponding to $\boldsymbol{\mu}(C_i\vartheta, e)$, $\boldsymbol{\Sigma}(C_i\vartheta, e)$ and the number of samples $|\mathcal{S}(C_i\vartheta, e)|$ for all substitutions ϑ grounding only the input argument of C_i such that $e \models C_i\vartheta$.

Observation 2 *Let* $\varphi \subseteq \psi$ *be features and let* $e = (H, \boldsymbol{\theta})$ *be an example. Let* $\vartheta : inp(\varphi, \psi, \lhd) \to \mathcal{D}_T(\varphi, \psi, e, \lhd)$ *be a substitution. Then* $\boldsymbol{\mu}(\varphi\vartheta, e)$, $\boldsymbol{\Sigma}(\varphi\vartheta, e)$, $n_{\varphi\vartheta} = |\mathcal{S}(\varphi\vartheta, e)|$ *are contained in*

$$(inp(\varphi, \psi, \lhd)\vartheta, \boldsymbol{\mu}(\varphi\vartheta, e), \boldsymbol{\Sigma}(\varphi\vartheta, e), n, \gamma) \in \bigoplus_{\lhd}^{\varphi,\psi} \mu\sigma'(\varphi, \psi, e, \lhd, T)$$

(where T *is a table with pre-computed domains of sub-features of* φ*) provided that* $\mu\sigma'(\varphi, \psi, e, \lhd, T)$ *are correct sample parameters.*

References

1. Shandar Ahmad and Akinori Sarai. Moment-based prediction of dna-binding proteins. *Journal of Molecular Biology*, 341(1):65 – 71, 2004.
2. Carolyn J M Best et al. Molecular alterations in primary prostate cancer after androgen ablation therapy. *Clin Cancer Res*, 11(19 Pt 1):6823–34, 2005.
3. Nitin Bhardwaj, Robert E. Langlois, Guijun Zhao, and Hui Lu. Kernel-based machine learning protocol for predicting DNA-binding proteins. *Nucleic Acids Research*, 33(20):6486–6493.
4. Michael E Burczynski et al. Molecular classification of crohns disease and ulcerative colitis patients using transcriptional profiles in peripheral blood mononuclear cells. *J Mol Diagn*, 8(1):51–61, 2006.
5. Patricia L M Dahia et al. A hif1alpha regulatory loop links hypoxia and mitochondrial signals in pheochromocytomas. *PLoS Genet*, 1(1):72–80, 2005.
6. Pedro Domingos, Stanley Kok, Daniel Lowd, Hoifung Poon, Matthew Richardson, and Parag Singla. Probabilistic inductive logic programming. chapter Markov logic, pages 92–117. Springer-Verlag, 2008.

7. William A Freije et al. Gene expression profiling of gliomas strongly predicts survival. *Cancer Res*, 64(18):6503–10, 2004.
8. I Gashaw et al. Gene signatures of testicular seminoma with emphasis on expression of ets variant gene 4. *Cell Mol Life Sci*, 62(19-20):2359–68, 2005.
9. Gavin J Gordon. Transcriptional profiling of mesothelioma using microarrays. *Lung Cancer*, 49 Suppl 1:S99–S103, 2005.
10. Nicholas J. Higham. Computing the nearest correlation matrix - a problem from finance. *IMA Journal of Numerical Analysis*, pages 329–343, 2002.
11. Matěj Holec, Filip Železný, Jiří Kléma, and Jakub Tolar. Integrating multiple-platform expression data through gene set features. In *Proceedings of the 5th International Symposium on Bioinformatics Research and Applications*, ISBRA '09, pages 5–17. Springer-Verlag, 2009.
12. Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 433–440. ACM, 2009.
13. Michael Jakl, Reinhard Pichler, Stefan Rmmele, and Stefan Woltran. Fast counting with bounded treewidth. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *LNCS*, pages 436–450. Springer Berlin / Heidelberg, 2008.
14. M. Kanehisa, S. Goto, S. Kawashima, Y. Okuno, and M. Hattori. The kegg resource for deciphering the genome. *Nucleic Acids Research*, 1, 2004.
15. M A Kuriakose et al. Selection and validation of differentially expressed genes in head and neck cancer. *Cell Mol Life Sci*, 61(11):1372–83, 2004.
16. Ondřej Kuželka and Filip Železný. Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning, online first, DOI: 10.1007 / s10994-010-5208-5*, 2010.
17. Clemens R Scherzer et al. Molecular markers of early parkinsons disease based on gene expression in blood. *Proc Natl Acad Sci U S A*, 104(3):955–60, 2007.
18. Eric W. Stawiski, Lydia M. Gregoret, and Yael Mandel-Gutfreund. Annotating nucleic acid-binding function based on protein structure. *Journal of Molecular Biology*, 326(4):1065 – 1079, 2003.
19. András Szilágyi and Jeffrey Skolnick. Efficient prediction of nucleic acid binding function from low-resolution protein structures. *Journal of Molecular Biology*, 358(3):922 – 933, 2006.
20. Yuko Tsuchiya, Kengo Kinoshita, and Haruki Nakamura. Structure-based prediction of dna-binding sites on proteins using the empirical preference of electrostatic potential and the shape of molecular surfaces. *Proteins: Structure, Function, and Bioinformatics*, 55(4):885–894, 2004.
21. Jue Wang and Pedro Domingos. Hybrid markov logic networks. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. AAAI Press, 2008.
22. M. Yannakakis. Algorithms for acyclic database schemes. In *International Conference on Very Large Data Bases (VLDB '81)*, pages 82–94, 1981.