Fisher Kernel for Tree Structured Data

Luca Nicotra
Department of Computer Science
University of Pisa
Pisa, Italy
E-mail: nicotra@cli.di.unipi.it

Alessio Micheli
Department of Computer Science
University of Pisa
Pisa, Italy
E-mail: micheli@di.unipi.it

Antonina Starita
Department of Computer Science
University of Pisa
Pisa, Italy
E-mail: starita@di.unipi.it

***Abstract*— We introduce a kernel for structured data, which is an extension of the Fisher Kernel used for sequences [11]. In our approach, we extract the Fisher score vectors from a Bayesian Network, specifically a Hidden Tree Markov Model [6], which can be constructed starting from the training data. Experiments on a QSPR (quantitative structure-property relationship) analysis, where instances are naturally represented as trees, allow a first test of the approach.**

I. Introduction

Kernel functions defined on structured data are receiving more and more attention, as a way of dealing with many "real-world" learning problems, such as Bioinformatics, NLP, or Document Processing. On one side they exploit the successful Kernel Methods (e.g. Support Vector Machines (SVMs)), to handle directly data naturally represented by graphs, but on the other, they are often designed a priori or at least they allow a limited adaptation, so that they cannot grasp any structural property that has not been guessed by the designer. Some examples are convolution kernels [4] which recursively take into account subgraphs, string kernels [18] which require that each tree is represented by the sequence of labels generated by a depth-first traversal of the trees, or graph kernels [8] based on a measure of the walks in two graphs that have some labels in common.

Another approach consists in adaptive kernel functions, which, at a certain computational cost, are able to adapt the kernel to the dataset. This is done mainly by the use of kernel functions defined on probabilistic, generative models, and by now the most used generative models have been Hidden Markov Models [10] with some exceptions, e.g. Probabilistic Hierarchical Models [17].

The model we develop belongs to this second framework. We extend the Fisher Kernel [11] previously applied to sequential data, to deal with trees, by using Hidden Recursive Models (HRMs) [7] as the model from which the kernel is extracted. HRMs allow to deal with varying structure data, defining recursively the Bayesian Network from which the data are supposed to be generated.

Actually, mainly for computational reasons, as we will explain in the subsequent sections, in the experiments we work with Hidden Tree Markov Models (HTMMs) [6]. They represent a subset of HRMs to deal with trees and we modify them adding Gaussian nodes to represent the target values. With this combination of generative and discriminative models, we can easily use the statistics extracted by a Bayesian Network for continuous regression tasks, which is not always easy, or natural, directly with a Bayesian Network.

Since our focus is on the adaptive processing of structured data, in the experimental part we also considered Recursive Neural Network models [16], which are able to learn a direct mapping between a structured domain and the output space.

The outline of the paper is as follows: first we introduce and characterize Hidden Recursive Models and Hidden Tree Markov Models. In the subsequent sections we review the basic concepts about Kernel Methods and Support Vector Regression in particular. Then we introduce the Tree Fisher Kernel as an extension of the one used with sequential data. Finally we conclude with results on preliminary experiments with a regression task in Cheminformatics.

II. Methods

A. Domain Description

We consider the input space of rooted positional k-ary trees T, with a label associated to each node. The labels and structure of the trees are supposed to obey an unknown probability distribution.

B. Hidden Recursive Models

Hidden Recursive Models (HRMs, [7]) are a class of probabilistic models for structure processing. Instead of defining the whole Bayesian Network from which the data are supposed to be generated, we define the so called *recursive network* (Figure 1), a pattern of hidden and observed nodes that is unrolled through each tree from the input space to generate the *encoding network* (Figure 2), a Bayesian network in which the conditional probability tables are shared among the replicas of the basic recursive network. So, although the description on the model is fixed, the topology of the encoding network (the Bayesian network on which inference must be performed) changes with each training example.

This is only the simplest case, in which a full stationarity through the network is supposed, but more complex encoding schemas can be used, (e.g a level-wise stationarity) even if they are hardly exploited because they tend to increase the model complexity too much.

Different problems arise with HRMs. In fact, the varying structure implies that, when inference must be performed, a different junction tree needs to be constructed for each

training example. This can be very computationally costly, so one solution can be to merge the training examples into an optimally compressed supergraph, as suggested in [7]. We decided instead to restrict ourselves to a more tractable class of models, the so called Hidden Tree Markov Models (HTMMs). This architecture was first suggested in [7] and developed in details in [6] for document processing. Please refer to those papers for a comprehensive description.

The global structure of an HTMM can be thought as divided into two identical skeletons **X** and **Y**. Nodes of **X**, denoted by $X(v)$, are labelled by hidden state variables, whether nodes of **Y** are labelled by the observed variables $Y(v)$, that is the data example.

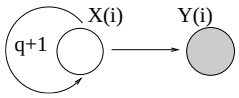

Fig. 1. Recursive Network of a Hidden Recursive Model. q^{+1} is an operator which represent a connection with the children of a given node

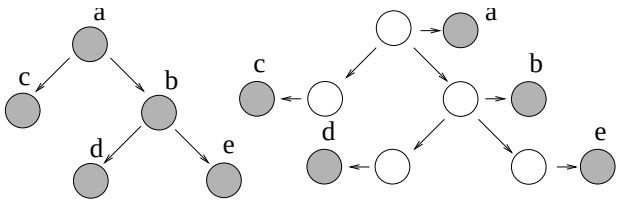

Fig. 2. A tree and the corresponding Encoding Network for a Hidden Tree Markov Model. Shaded variables receive evidence during learning

For this model it holds a condition similar to the one that holds for Hidden Markov Models (HMMs), called the first-order tree-Markov property if the following conditions are met: the first-order tree-Markov property must hold for the hidden tree X, and ($\forall v$) observation $Y(v)$ is independent of the rest given $X(v)$. These conditions imply the following global factorization formula:

$$P(\mathbf{Y},\mathbf{X}) = P(X(r))P(Y(r)|X(r)) \cdot \cdot \prod_{v\in V\setminus r} P(Y(v)|X(v))P(X(v)|X(pa[v])) \quad (1)$$

where V is the set of all the nodes of the tree, v indicates a generic node, $pa[v]$ the parent node of v and r represent the root node. This formula can be represented by a Bayesian network as shown in (Figure 2).

Accordingly the parameters of the network are $P(X(r))$, the prior on the root hidden state, $P(Y(v)|X(v))$, the emission conditional probability table, and $P(X(v)|X(pa[v]))$, the hidden conditional probability table.

With this simple structure a specialized version of the JLO algorithm [12] can be used for inference as explained in [7]. In particular no moralization is required and the cliques are all formed by two variables (two hidden variables or a hidden and an observed variable).

We use Expectation Maximization (EM) as learning algorithm [14]. This, in a Bayesian Networks (and thus also in HTMMs), is performed by first computing the expected sufficient statistics for the parameters and then updating parameters with the normalized sufficient statistics. For an explicit equation formulation refer to [6].

C. Conditional Gaussian Distributions

In order to further help and influence the capability of the model in extracting the relevant features for the regression task that will be presented in the experiment section, we add a Gaussian node (with an associated Gaussian variable) to the recursive network, representing the target real value (Figure 3). In this way, every hidden node of the resulting network has a Gaussian child (Figure 4). In other words this builds a third skeleton identical to **X** and **Y** which can be associated to the continuous nodes. The resulting network is a mixed Bayesian Network with conditional Gaussian distribution.

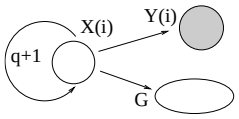

Fig. 3. Recursive Network of a Hidden Recursive Model with Gaussian nodes (represented as ellipses). q^{+1} is an operator which represent a connection with the children of a given node

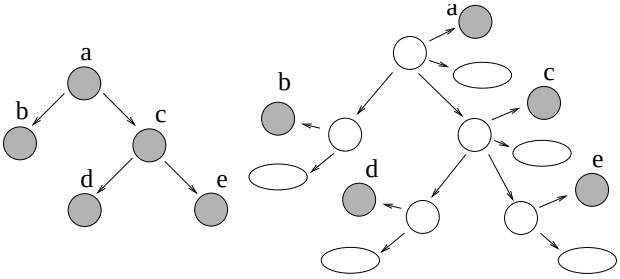

Fig. 4. A tree and the corresponding Encoding Network for a Hidden Tree Markov Model with Gaussian nodes (represented as ellipses). Both Gaussian and shaded discrete variables receive evidence during learning

The conditional distributions of discrete variables given their (discrete) parent variables are specified as usual, whereas the conditional distribution of continuous variables are assumed to be Gaussian, i.e.

$$P(Y(v)|X(v)=i) = \boldsymbol{N}(\mu,\sigma^2) \quad (2)$$

whenever $p(i) = P(X = i) > 0$, where Y denotes the continuous variables, X the discrete parent, and i its hidden state. We then say that $X \cup Y$ follow a Conditional Gaussian (CG) distribution.

It must be noted that a different univariate Gaussian is associated to each of the states of the hidden parent whose probability is not 0, and for each Gaussian the mean parameter μ is estimated (while the variance σ^2 is fixed to be 1).

Usually continuous nodes add a certain degree of complexity, but given that in our model they do not have any discrete child, we can use a simplified version of the inference algorithm proposed in [13]. This algorithm, which is a modified version of the general junction tree algorithm, do not suffer the instability problems which affected the previous learning algorithms for CG distributions.

A theoretical motivation for this structure comes from Jaakkola [11] who proves that "*a kernel classifier employing the Fisher kernel derived from a model that contains the label (that is the target variable) as a latent variable is, asymptotically, at least as good a classifier as the MAP labeling based on the model*".

Other motivations will be introduced after the description of the domain to which our model will be applied.

In the subsequent two subsections we turn to the description of the discriminative part of the model, directly focusing on the kernel method we use in our implementation.

D. Support Vector Regression

Regression estimation is concerned with estimating real-valued functions. For this task an analog of the soft margin can be constructed in the space of the target values by using Vapnik's ϵ-insensitive loss function [15]. Essentially a tube with radius ϵ is fitted to the data. The trade-off between model complexity and points lying outside of the tube can be found solving a quadratic programming problem. Finally regression estimate takes the form

$$f(x) = \sum_{i=1}^{m} (\alpha_i^* - \alpha_i) K(x_i, x) + b \tag{3}$$

where α_i^* and α_i are non negative and solve the quadratic programming problem, while b is a constant. This is called Support Vector Expansion.

The contribution of each sample (tree) to the decision rule consists of two parts: 1) the overall importance of the training example X_i as summarized with the non-negative coefficients α_i^*, α_i and 2) a measure of pairwise "similarity" between the training example X_i and the new example X, expressed in terms of a kernel function $K(X_i, X)$, which may vary with different domain. In fact, we introduce in the following section the Tree Fisher Kernel to deal with structured examples.

III. Tree Fisher Kernel

To combine an HTMM with a discriminative SVM we use an extended version of the previously developed Fisher Kernel. We refer to it as the Tree Fisher Kernel.

As we showed before, the HTMM assigns a probability to any given tree of the dataset. But, if we want to compare trees with different topologies we need a uniform (in the number of features) representation for each tree.

In the domain of sequences it has been shown that the gradient of the log-likelihood with respect to the parameters is an effective metric [11].

So, similarly, for each tree we extract a representation in the form of what are known as sufficient statistics. Actually we don't work directly with the vector of the sufficient statistics but with an analogous quantity known as the Fisher score:

$$U_X = \nabla_\theta \log P(X|\theta) \tag{4}$$

where θ is the vector of the parameter of the model.

It can be shown that the natural kernel of this mapping is the inner product between these feature vectors relative to the local Riemannian metric:

$$K(X, X') = U_X^T I^{-1} U_{X'} \tag{5}$$

where I is the information matrix, but it is less significant and it is usually approximated to an identity matrix.

The gradient of the log-likelihood with respect to a parameter in (4) describes how the parameters contribute to the process of generating a particular example, so the Fisher Kernel measures distances in the space of the respective probabilistic model parameters (gradient space of the generative model). Moreover it can be shown that it preserves all the structural assumptions of the model from which it is extracted (that is, the mutual dependencies between the variables of the model).

While in the context of HMMs these statistics are computed with the application of the standard forward-backward algorithm, with HTMMs with continuous nodes we can't apply the same algorithm. Nonetheless, we can similarly use the JLO algorithm modified to deal with continuous nodes [13] to extract the posterior expectations used to compute the Fisher scores and thus the cost of computing the Fisher score vectors is of the same order as simply evaluating $P(X|\theta)$.

With these fixed length gradient vectors U_X and $U_{X'}$ we can now find an appropriate kernel function to quantify their similarity. For this purpose one of the most common kernel functions can be used. Some examples are radial basis function kernels or sigmoidal kernel.

To summarize, we estimate the parameters of a Hidden Tree Markov model trained from examples of trees. Then we use this Hidden Tree Markov Model to map each tree X into a fixed length vector, its Fisher score, and compute the kernel function on the basis of the distance between the score vector of the tree and the score vectors for known trees of the tree dataset. Finally, a SVM is used to realize the regression model.

IV. Application to QSPR analysis

We applied the Tree Fisher Kernel to QSPR (Quantitative Structure-Property Relationships) analysis, i.e. predicting the boiling point of a group of alkanes, to test the quality of the parameters extracted by the Tree Fisher Kernel. This is a regression problem on a structured domain involving chemical compounds represented as trees and it has been used in testing various QSPR approaches [2], [3].

It is well known from QSPR analysis that, for this problem, there is a clear correlation between molecular shape and

boiling point, and that the target property is related to global characteristics of the structures, such as the molecular size and the molecular shape.

Here, this task has been selected in order to have a direct comparison of the new approach based on Tree Fisher Kernel versus Recursive Neural Network methods [16] in a real-world application. Note that, the comparison is meaningful since both approaches directly deal with tree representation of data and allow adaptive computation of the features representing the data. In particular, Recursive Neural Networks allow a general approach to the adaptive processing of structured data, since such a discriminative model is able to automatically encode the structural information depending on training data and on the computational problem at hand. A Recursive Neural Networks approach, based on Recursive Cascade Correlation (RCC) model [2], has been proved to be competitive with respect to *ad-hoc* techniques at the state-of-art in the field. In particular, the results have been compared with the approach reported in [3], which yields very good performances applying a multilayered feedforward neural network to a vectorial representation of alkanes able to retain the structural information which is known to be relevant to the prediction of the boiling point.

The data set that composes our benchmark, which is taken from [3], is based on all the 150 alkanes with up to 10 carbon atoms (C_nH_{2n+2}). It is well known that for this class of compounds, the prediction of the boiling point can be performed by disregarding the information about the hydrogen atoms. Hydrogens suppressed graphs of alkane molecules are trees. Carbon-hydrogen groups are associated with vertexes, and bonds between carbon atoms are represented by edges. In order to uniquely represent them as rooted positional trees, we used the standard conventions used in chemistry, i.e. the I.U.P.A.C. nomenclature rules [9]. The details of the definition of appropriate rules for the specific set of molecules studied, are discussed in [2]. The prediction of the boiling point yields to a regression task with a target associated to the root vertex of each tree. The target is the boiling point expressed in Celsius degrees (^{0}C) into the range $[-164, 174]$.

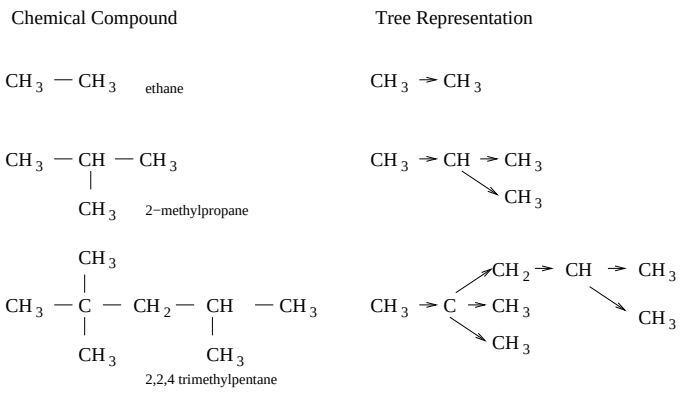

Fig. 5. Alkanes representation

With this representation we can use a Hidden Tree Markov Model to process the structure of these chemical compounds in the form of labelled trees. Although several forms of stationarity can be used, we choose a fully stationary HTMM (using the same condititional probability table in every node of the network) to keep the number of free parameters reasonable. In fact, the small number of compounds that is usually available for this kind of analysis can easily result in overfitting, if we do not take into account the complexity of the model. The drawback of this approach is that we neglect certain structural aspects. For example we do not consider the position of a child node (left, center, right) but we could, using for example, different forms of stationarity, e.g. different conditional probability table for every kind of child. In order to further decrease the number of parameters, and at the same time, to put some prior on the parameters, we used conditionally probability tables without cycles except for the self loops. HMMs with this property are generally referred to as left-right (or sometimes Bakis models) but we prefer to use the top-down terminology, which better suits our tree-shaped model and data. This matrix structure seemed to be very plausible in the alkane domain, in which the depth of the tree appears to be correlated with the target boiling point value.

A consequence of this choice is that, in our case, not only the magnitude of the Fisher score components specify the extent to which each parameter contributes to generating the tree from which they are extracted, but they assume an even more precise semantic because every parameter can be used (and, as a consequence, its Fisher score component can be different from zero) in the nodes only from a specific depth.

Then we faced the following problem: how can we force the model to distinguish between two structures that share the same prefix but whose target values are very different?

We decided to introduce the target values in the model as previously described. This should be sufficient, because, to increase its likelihood each tree tries to use the Gaussians whose means are near its target value. So, although their observed values might be the same and the two compounds could be partially parameterized by the same parameters, this tends to define a different pattern of "activation" of the parameters.

This structure is mainly motivated by the following use in a discriminative model, and the multiple variable representing the same target value, would probably be difficult to use for inference directly on the Bayesian Network.

This is an important point. Because we are using our Bayesian Network as a sort of feature extractor, we are really interested in making the algorithm learn the right parameters for our final task, not only in its ability to explain the data.

To show how this model structure influences the learned parameters, we propose an example

Example: Suppose we have two trees A and B, with respectively two and three nodes. The structure and the Encoding Network for both of them is shown in Figure 6.

Suppose that the first two nodes of the two trees share the same observations, but that their targets are quite different (-1

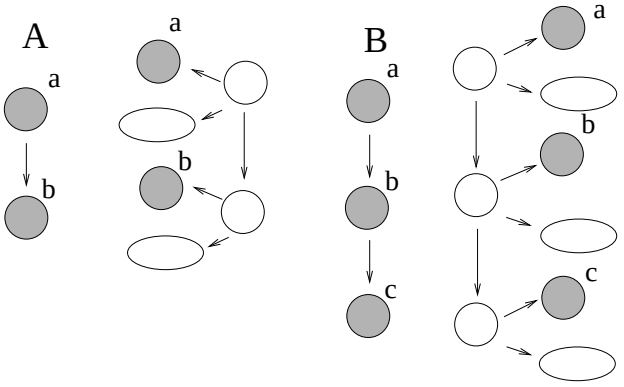

Fig. 6. Trees and corresponding Encoding Network for the trees in the example

for A and +1 for B). Moreover,for simplicity, each node can be in one of two states and thus the target Gaussian node is a mixture of two Gaussians (each corresponding to a hidden state) whose mean are -0.8 (corresponding to state 1) and 1.3 (corresponding to state 2). When the evidence about the target nodes is incorporated into the network, the probability of the two hidden states is subject to the following scaling:

$$p^*(i) = \frac{p(i)}{\sqrt{2\Pi}\sigma} e^{-\frac{1}{2}(y-\mu(i))^2/\sigma^2} \tag{6}$$

where $p(i)$ is the probability of state i, y is the target value, and $\mu(i)$ is the mean of the Gaussian variable corresponding to state i. For A this results in a scaling 0.39 of the probability of state 1 and 0.028 for state 2, while for B the scaling is 0.080 for state 1 and 0.38 for state 2. Hence we condition the A's likelihood to rely more on state 1 and B's likelihood on state 2. Similarly this influences the statistics extracted by the EM algorithm, hopefully defining different pattern of "activation" of the model parameters in the two cases. Actually the conditioning is a little more complex, because we should take in account the CG potential as explained in [13], but this goes beyond the intent of this example. The Fisher scores extracted from the model should differ substantially and the Fisher Kernel can take advantage of this.

Without the Gaussian target nodes, the algorithm can discriminate between A and B only relying on the last node of B, but it would probably not penalize a similar activation of the parameters in the first two nodes.

Furthermore, if we try to use only a target Gaussian node for the whole network, we can incur in a problem of insufficient "diffusion of credit"[1], and then only the nodes near the target would be strongly conditioned. But we haven't tested in which proportion this is a problem, and cannot make a clear statement.

A. Overview of experiments

Despite these considerations, the task on which we tested our model is quite difficult for different reasons. First, the dataset is really small, comprising only 150 examples, at the limit of statistical significance. In these conditions statistical models are known to perform poorly, much worse then, for example, neural networks. But the task seemed to be appropriated since one of the goals of our mixed discriminative-generative approach is trying to overcome these kind of problems.

We tested both the Fisher Kernel derived from the original HTMM and the one derived by adding the target continuous nodes. In this way we are able to evaluate the effectiveness of the new structure.

A radial basis function kernel with a value of σ equal to 0.5 was used as the metric to compare the different Fisher scores, while the C parameter which represents the trade off between error and margin in the SVM was fixed equal to 100.

We used a 10-fold cross validation (15 compounds for each fold) as in the original experiments [2].

In these experiments we utilized an implementation of the SVM developed by Collobert & al.[5], together with our implementation of the HTMM and the Fisher score vectors extraction.

Here we provide a comparison of the results of the various approaches. In Table 1 we give the performance of the Tree Fisher Score methods (TFK-CG refers to the version with Gaussian variables) compared with the Recursive Cascade Correlation (RCC) employed in [2] and the ad-hoc Multilayered Perceptron employed in [3].

For each model the mean absolute error and the maximum absolute error are reported.

TABLE I
EXPERIMENTAL RESULTS

Model	Mean abs. error	Max abs. error
Best MLP	3.01	10.42
Best RCC	2.74	13.27
Mean RCC	3.71	30.33
Best TFK	6.21	37.27
Mean TFK	7.81	50.32
Best TFK-CG	5.94	32.25
Mean TFK-CG	7.11	45.67

Our model is sensitive on the initialization values, and when the EM becomes stuck in a local minimum, the Fisher Kernel performs poorly.

In general the results for the Tree Fisher Kernel seem to be worse with respect to the one obtained by the previous approaches, in particular by the Recursive Cascade Correlation, and somehow confirm the difficulty of a statistical approach when applied to a very limited dataset.

However these results are still very preliminary, and if we consider that the setting of the model against which we compare the Tree Fisher Kernel has been fully explored against this regression task, they can be interpretated as promising.

In fact, our main motivation was to assess the capability of the Fisher Kernel to deal with trees (and the results seems to

confirm this), and not already to compete with the state-of-art algorithm for this task.

Only few configurations have been tested, and we believe that a clear analysis of the behavior of the Tree Fisher Kernel with the different trees can suggest the main lack of the current one.

Nonetheless the approach based on the network with Gaussian nodes performs better than the standard HTMM.

Among the other variations that can be tested on this problem there are: the number of hidden states, the characteristics of the conditional probability table, different stationarity patterns or the application of leave-one-out to increase the significance of the training set.

Moreover sometimes only a subset of the Fisher score entries can be significant for a problem, and for example in [11] only the gradient respect to the parameters of the observed variables are used.

V. Conclusions

We have developed a new kernel to deal with structured hierarchical data, extending both the Fisher kernel to trees, and the the HTMM to make use of discriminative models. The approach provides an interesting way to adaptively construct kernels for structured data considering the process that generated the data.

Although many structural property were not taken into account, we obtained promising results with this relatively simple model. We plan to further extend our model in order to be able to consider some additional information about the processed trees, for instance by the use of different stationarity schemas, or a more complex prior on the parameters.

One promising direction is the use of conditional models (that is, in practice, reversing the orientation of the edge between hidden and observed nodes). This will considerably increase the number of free parameters, and so the alkane dataset may not be suited for this approach.

Furthermore different combinations of the scores obtained with different model settings or different models can be used. The problem is that there is no clearly optimal way to combine these scores in practice.

Acknowledgment

The authors would like to thank Prof. Paolo Frasconi for helpful advice.

References

[1] Y. Bengio and P. Frasconi. Diffusion of Credit in Markovian Models. In G. Tesauro, D. Touretzky and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 553–560. MIT Press, 1995.

[2] A.M. Biancucci, A. Micheli, A. Sperduti, and A. Starita. Application of Cascade Correlation Networks for Structures to chemistry. *Applied Intelligence Journal*, 12(1/2):117–146, 2000.

[3] D. Cherqaoui and D. Villemin. Use of neural network to determine the boiling point of alkanes. *J. Chem. Soc. Faraday Trans.*, 90(1):97–102, 1994.

[4] M. Collins and N. Duffy. Convolution kernels for natural language. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

[5] R. Collobert and S. Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*,1:143–160, 2001.

[6] M. Diligenti, P. Frasconi, and M. Gori. Hidden Tree Markov Models for document image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):519–523, 2003.

[7] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786, 1998.

[8] T. Gärtner, A. P. Flach, and Wrobel S. On graph kernels and Gaussian processes for relational reinforcement Learning, 2003.

[9] IUPAC. *Nomenclature of Organic Chemistry*. Pergamon Press, Oxford, 1979.

[10] T. Jaakkola, M Diekhans, and D. Haussler. A Discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1,2):95–114, 2000.

[11] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 487–493. MIT Press, 1999.

[12] F. V. Jensen, S. L. Lauritzen, and Olsen K. G. Bayesian updating in recursive graphical models by local computations. 15:269–282, 1990.

[13] S. Lauritzen and F. Jensen. Stable local computation with conditional Gaussian distributions. *Statistics and Computing*, 11:191–203, 2001.

[14] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition, 1989.

[15] A. Smola and B. Scholkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2, 1998.

[16] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[17] A. Vinokourov and M. Girolami. A probabilistic framework for the hierarchic organisation and classification of document collections. *Information Processing and Management*, 2002.

[18] S. Vishwanathan and A. Smola. Fast kernels for string and tree matching. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.