Markov Logic Networks for Supervised, Unsupervised and Semisupervised Learning of Classifiers

Victor A. Silva and Fabio G. Cozman

Escola Politécnica – Universidade de São Paulo
Av. Prof Mello de Morais, 2231, CEP 05508-900
São Paulo, SP, Brazil
www.pmr.poli.usp.br/ltd/

Abstract. This paper investigates the application of *Markov logic networks* to supervised/unsupervised/semisupervised learning of classifiers. We review the necessary theory, propose techniques so as to represent the relevant learning problems in Markov logic, and present initial results in supervised learning with real data.

Key words: Markov logic networks, classification, machine learning.

Victor A. Silva is a MSc student with expected thesis defense in September 2008. This paper reflects his current work on machine learning and Markov logic.

1 Introduction

When learning a classifier, a central step is the selection of a language in which to express the classifier and the associated learning task. Some classifiers in the literature are based on the language of probabilistic graphical models, such as naive Bayes and decision trees classifiers. Other classifiers have their roots in logical languages — for instance, classifiers modeled within the inductive logic programming paradigm.

Recently, the learning community has stressed the need for languages that can unify statistical and relational aspects of various learning problems. As a result, a new research area named "statistical relational learning" has received great attention. One of the most promising tools that were created within this new area is *Markov logic* [1]. The purpose of Markov logic is to allow structured descriptions of observed data and also learning from this data. This is done by encoding dependence relations using first-order logic in such a way that any set of sentences represents a Markov network (that is, a Markov random field). Markov logic networks are very flexible tools that allow one to handle many aspects of real-world problems [2]. Learning algorithms have been implemented in an open tool called *Alchemy package* [3]. Markov logic is currently a natural choice if one wishes to combine relational fragments of first-order logic with probabilities.

In this work, we investigate Markov logic as a language for learning classifiers, as the behavior of Markov logic in classifier learning tasks has not been explored in detail in the literature. We focus on three main paradigms of machine learning. At first, supervised learning is considered, where a classfier is learned from a set of labeled instances [4, 5]. We present results with real data in supervised contexts. We then discuss some important aspects of how unsupervised/semisupervised leaning tasks could be modeled with Markov logic. We pay special attention to *semisupervised learning with constraints*, as this kind of learning task seems particularly amenable to the resources of Markov logic. This part of the paper is more speculative, describing work currently underway.

Section 2 reviews important concepts on learning paradigms and Markov logic. In Section 3 we present our main ideas on the use of Markov logic in learning. We also describe experiments on real data in supervised problems.

2 Background

This section defines a few basic concepts. First, we introduce the main ideas in classification and machine learning that we need; second, we present Markov logic networks.

2.1 Supervised, unsupervised, and semisupervised learning of classifiers

Methods usually employed in machine learning can be partitioned in two main groups: *supervised* and *unsupervised* methods [6]. Those methods focusing on supervised learning (SL) learn a mapping (a classifier) from attributes X to class label Y, given a training dataset with pairs (x_i, y_i). Classifiers learned can have their perfomance evaluated using test datasets. As an example, spam detection is often modeled as a supervised learning problem.

In unsupervised learning (UL), also known as *clustering*, the goal is to find interesting structures in an observed dataset, represented usually by a vector of measurements $[x_1, \ldots, x_n]$. In other words, clustering techniques try to organize a collection of data into clusters based on similarity. In an intuitive manner, observed values within a valid cluster are more similar to each other than they are alike a data point in a different cluster.

Recently, learning tasks between SL and UL have received attention, under the name *semi-supervised learning* (SSL). In this paradigm, the dataset available for learning tasks contains, in addition to unlabeled data, some supervised information. In many applications, this information is the class label associated with some observations. In the contexts where SSL takes place, a dataset can be split into two pieces: the observations $X_l = [x_1, \ldots, x_l]$ for which labels $Y_l = [y_1, \ldots, y_l]$ are given, and the observations $X_u = [x_{l+1}, \ldots, x_{l+u}]$ whose labels are unknown. Providing labels for some observations is not the only way to introduce supervised information. This can also be done, for example, by establishing constraints such as data points that have (or do not have) the same class label.

Indeed, in recent years, the problem of learning with constraints has been addressed by a considerable number of researchers. Some applications for intelligent systems generate natural pairwise constraints (e.g., image segmentation and video and document retrieval). Moreover, in an interactive setting a user who is not a domain expert may sometimes provide feedback in the form of pairwise constraints more easily than class labels, since providing constraints does not require the user to have significant prior knowledge about the categories in the dataset. Usually, the literature uses the terminology *must-link constraint* and *cannot-link constraint* to specify points that should or should not be clustered together, respectively.

A few previous results in semisupervised learning with constraints deserve to be mentioned. Law et al [7] employ a graphical model in model-based clustering with soft and probabilistic constraints. The uncertainty of a constraint is encoded by the distribution of the constraint random variables. The value of each constraint reflects the uncertainty of the prior knowledge that a pair of points comes from the same class. They show that more robust clustering solutions can be obtained by allowing uncertainty in the model. Shental et al [8] focus is on semi-supervised learning using *side-information*, which is not given as labels, but as "is-equivalent" and "not-equivalent" constraints, similarly to must and cannot-links. They show how to incorporate equivalence constraints into the EM algorithm [9], in order to compute a generative Gaussian mixture model of the data. Basu et al [10] introduce an approach to semisupervised clustering based on hidden Markov random fields that combines constraint-based and distance-based approaches in a single probabilistic model. Typically, the constraints are also soft, that is, clusterings that violate them are undesirable but not prohibited.

2.2 Markov logic networks

A *first-order knowledge base* (KB) is a set of sentences or formulas in first-order logic [11]. Formulas are constructed using four types of symbols: constants (e.g., domain of people: `Anna`, `Bob`), logical variables ranging over objects of a domain on interest, functions (e.g., `MotherOf`) representing mappings from tuples of objects to objects, and predicates representing relations among objects in the domain (e.g., `Friends`) or attributes of objects (e.g., `Smokes`). If a world violates even one formula, it has probability zero. A KB can thus be interpreted as a set of hard constraints on the set of possible worlds. Markov logic networks soften these constraints so that when a world violates one formula in the KB it becomes less probable, not impossible. The fewer formulas a world violates, the more probable it is.

A *Markov logic network* (MLN) [1] is defined as a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, \ldots, c_{|C|}\}$, it defines a Markov network (MN) $M_{L,C}$ as follows.

1. $M_{L,C}$ contains one binary node for each possible grounding[1] of each predicate appearing in L. The value of the node is 1 if the ground predicate is true and 0 otherwise.
2. $M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature, which reflects the strength a constraint, is the w_i associated with the F_i in L.

There is an edge between two nodes of $M_{L,C}$ if the corresponding ground predicates appear together in at least one grounding of one formula in L. The probability distribution over possible worlds x specified by the ground MN $M_{L,C}$ is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i}^{F} w_i n_i(x)\right),$$

where F is the number of formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN converges to a purely logical KB, being equal to one in the limit of all infinite weights. Assuming domain closure and finiteness, we have that propositionlized MLNs are finite. In this case, the groundings of a formula are built simply by replacing its variables with constants in all possible combinations.

As a didactic example to elucidate the use of MLN, consider a hypothetical classification task involving three attributes (features): two of them being predictors and the third one being the class label attribute, on which predictions should be made. Assume the following first-order formulas that fit a model to a database available for training a classifier are given:

$\forall \mathtt{x}\ \mathtt{Feature}_1(\mathtt{x}) \Rightarrow \mathtt{Class}(\mathtt{x})$,
$\forall \mathtt{x}\ \mathtt{Feature}_2(\mathtt{x}) \Rightarrow \mathtt{Class}(\mathtt{x})$,
$\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Relation}(\mathtt{x},\mathtt{y}) \Rightarrow \mathtt{Feature}_2(\mathtt{x}) \Leftrightarrow \mathtt{Feature}_2(\mathtt{y})$.

If we have the constants $C = \{\mathtt{A}, \mathtt{B}\}$, the MLN L yields the following ground formulas in the grounded MN $M_{L,C}$:

$\mathtt{Feature}_1(\mathtt{A}) \Rightarrow \mathtt{Class}(\mathtt{A})$
$\mathtt{Feature}_2(\mathtt{A}) \Rightarrow \mathtt{Class}(\mathtt{A})$
$\mathtt{Feature}_1(\mathtt{B}) \Rightarrow \mathtt{Class}(\mathtt{B})$
$\mathtt{Feature}_2(\mathtt{B}) \Rightarrow \mathtt{Class}(\mathtt{B})$
$\mathtt{Relation}(\mathtt{A},\mathtt{B}) \Rightarrow \mathtt{Feature}_2(\mathtt{A}) \Leftrightarrow \mathtt{Feature}_2(\mathtt{B})$
$\mathtt{Relation}(\mathtt{B},\mathtt{A}) \Rightarrow \mathtt{Feature}_2(\mathtt{B}) \Leftrightarrow \mathtt{Feature}_2(\mathtt{A})$
$\mathtt{Relation}(\mathtt{A},\mathtt{A}) \Rightarrow \mathtt{Feature}_2(\mathtt{A}) \Leftrightarrow \mathtt{Feature}_2(\mathtt{A})$
$\mathtt{Relation}(\mathtt{B},\mathtt{B}) \Rightarrow \mathtt{Feature}_2(\mathtt{B}) \Leftrightarrow \mathtt{Feature}_2(\mathtt{B})$.

From these grounded formulas, a proposicionalized Markov network is built. Figure 1 depicts it, where $\mathtt{C(U)}$ stands for $\mathtt{Class(U)}$, $\mathtt{F_j}(\mathtt{U})$ stands for $\mathtt{Feature_j(U)}$ and $\mathtt{R(U,V)}$ for $\mathtt{Relation(U,V)}$.

[1] The term "grounding" means an instanciation of the predicate or function, i.e., the replacement of a variable by its possible constant values.

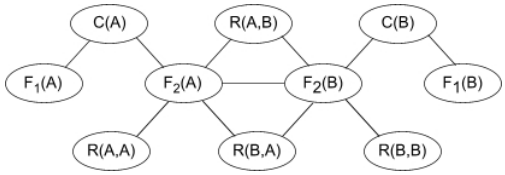

Fig. 1. Propositionalized Markov network for a set of constants $C = \{\mathtt{A},\ \mathtt{B}\}$.

Weight learning in MLNs can be done in a discriminative or generative approach. Usually, it is carried out by the maximization of a likelihood function of the state of a clique in the corresponding MN. To perform probabilistic inference, algorithms based on MCMC methods are usually employed (the Gibbs sampler is commonly used). Details on MLNs can be found in [12, 13].

3 Learning with Markov Logic Networks

In this section we show how to use MLNs in learning from data. Section 3.1 presents parcial results for SL tasks with real data. Section 3.2 briefly discusses unsupervised and semisupervised problems, indicating the path our current work is following.

3.1 Supervised learning scenarios

One might consider learning classifiers by learning weights for all formulas such as

$$\mathtt{C} \Rightarrow \mathtt{A_1} \wedge \ldots \wedge \mathtt{A_n},$$

where C is the class variable and A_i is an attribute. However, as n grows, this approach becomes untractable as it requires estimation of an exponential number of probability values. Indeed, if each atribute has m different values and we have c possible classes, $c \cdot m^n$ parameters must be learned from data. Overfitting then emerges and leads to poor classifiers.

A better alternative is to learn probabilities for Horn clauses that mimic the structure of a Naive Bayes classifier, such as

$$\mathtt{C} \Rightarrow \mathtt{A_1}, \ldots, \mathtt{C} \Rightarrow \mathtt{A_n},$$

whose goal then is to learn the probability that a class label value implies a specific value for each attribute A_i. We now have only $n\,(c \cdot m)$ parameters to estimate. Even with reduced number of parameters to determine, our preliminary experiments have shown bad performance with the resulting classifiers. Further analysis revealed it to be actually far from a Naive Bayes classifier, because weights of formulas like $\mathtt{C} \Rightarrow \mathtt{A_i}$ are not related at all with the conditional probability of $P(A_i|C)$. Indeed, the probability of an implication to be true tells us nothing about the conditional probabilities, as the formula is always true

if $\neg \mathtt{C} \vee \mathtt{A_i}$. Even though in [14] we found an approach to express conditional probabilities from weighted clauses, it does not seem possible to create sensible classifiers on the basis of learning probabilities for Horn clauses.

The best results we reached with MLNs have been produced instead with conjuntions of class and attributes, denoted by the following formulas:

$$\mathtt{C} \wedge \mathtt{A_1}, \ldots, \mathtt{C} \wedge \mathtt{A_n}.$$

The methodology adopted to express a classifier in Markov logic based on these formulas can be explained as follows.

1. Define the class label variable.
2. Convert attributes (predictors) and class label variable into representative first-order predicates.
3. Build conjunctions between pairs of class and attribute predicates.
4. Build the traning and testing databases.
5. Learn weights for formulas.
6. Cross-validate the classifier model obtained.

To illustrate these steps, consider the following example. A database usually contains rows with observed values for attributes and for class labels. Supposing a database contains a hundred rows with binary attributes, so a classifier representation for this example could be constructed in Alchemy syntax with three predicates

$$\texttt{Attribute\_i}(\texttt{row}, \texttt{value\_i!}), \quad \texttt{Attribute\_j}(\texttt{row}, \texttt{value\_j!}),$$

$$\texttt{Class}(\texttt{row}, \texttt{value\_c!}),$$

and two logic formulas:

$$\texttt{Class}(\texttt{row}, +\texttt{value\_i}) \wedge \texttt{Attribute\_i}(\texttt{row}, +\texttt{value\_i})$$

$$\texttt{Class}(\texttt{row}, +\texttt{value\_j}) \wedge \texttt{Attribute\_j}(\texttt{row}, +\texttt{value\_j}).$$

The variable `row` ranges over each row number in database, and `value_i`, `value_j` ranges over the values the respective attribute can assume. The operator `!` placed after a variable allows one to specify variables that have mutually exclusive and exhaustive values. For example, in `Class(row, value_c!)`, the `!` means that any row has exactly one class label value. If one or multiple variables are preceded by `+`, a weight is learned for each combination of their values in the ground formula.

To investigate the performance of Markov logic classifiers, we have performed experiments with some of the UCI repository (http://archive.ics.uci.edu/ml/) using the Alchemy package [3]. Table 1 summarizes the accuracy of our classifier, denoted by MLNC (Markov Logic Network Classifier), compared to other state-of-art classification methods. None attribute selection method was employed to any of the databases.

Table 1. Classification results (in %) for MLNC and other methods, obtained with ten-fold cross-validation. The results for the other classifiers were obtained from Weka (Waikato Environment for Knowledge Analysis) [15]. #Lab and #Att are the number of labeled instances and number of attributes (predictors + class), respectively. Information about standart deviation in classification accuracy results for Weka methods is not available.

Bases	Bases Info.		Rules			Trees			Bayes Nets		MLNC
	#Lab	#Att	ZeroR	DT	Prism	SC	DS	J48	NB	TAN	
Adult	30162	14+1	**75.11**	**85.36**	–	–	**75.10**	**85.73**	**82.84**	**85.98**	**86.77 ± 0.34**
Ionosphere	351	34+1	**64.10**	**87.18**	**84.90**	**88.60**	**82.90**	90.31	90.31	92.59	**89.76 ± 3.75**
Nursery	3087	8+1	**34.20**	96.00	99.52	99.58	**67.98**	98.07	92.49	96.45	**90.96 ± 1.13**
Breast Cancer	569	25+1	**62.74**	**94.20**	**92.27**	**95.25**	**90.51**	**95.43**	**95.61**	96.31	**95.79 ± 3.22**
Image Segmen.	210	19+1	**14.28**	83.81	87.62	93.33	**28.10**	93.81	89.05	95.71	75.24 ± 2.71
Car Evaluation	1728	6+1	**70.02**	91.49	89.70	97.11	**70.02**	92.36	**85.53**	94.61	**88.25 ± 6.26**
Balance Scale	609	2+1	**45.76**	**71.04**	**37.44**	**69.60**	**56.32**	**69.60**	**70.72**	**71.36**	**95.08 ± 3.18**
Iris	150	4+1	**33.33**	**94.00**	**92.67**	**94.00**	**66.67**	**94.00**	**94.00**	**94.67**	**95.33 ± 6.32**
Monk's	556	6+1	**48.39**	89.52	90.32	80.64	**73.39**	82.26	77.42	95.97	72.26 ± 7.12
Wine	178	13+1	**39.89**	**96.08**	**95.51**	**93.82**	**58.43**	**93.82**	98.87	98.31	**97.22 ± 5.40**

The values in boldface for MLNC classifier indicate notable accuracy results, and the boldface values for other classifiers indicate those whose accuracy value was surpassed by MLNC results. In comparison with the rule-based classifiers, MLNC presented superior accuracy values for most databases. Regarding tree-based classifiers, MLNC had better or very similar results in 6-out-of-10 databases. In comparison with Bayesian networks based classifiers such as Naive Bayes (NB) and Tree-Augmented Network (TAN), MLNC presented competitive results. In some cases MLNC surpassed the TAN classifiers, generally considered the best available Bayesian network based classifiers [16].

3.2 Unsupervised and semisupervised learning scenarios: a proposal

SL requires fully labeled patterns while learning a classifier. Labels are often specified using human expertise, which is expensive, time consuming and error prone [17]. Unlabeled data are usually easier to obtain. As an example, in facial image recognition it is easy to collect videos of people's expressions; however, it is expensive to label videos to the corresponding expressions.

Usually, UL methods cluster datasets based on a similarity measure, frequently represented as a distance function applied to a pair of observations [18]. Most times this measure is the only information available to allocate an observation into a particular cluster. Consequently, information about dependence relations "within" the data can be inadvertently discarded. Our idea is to encode domain knowledge about the data to be clustered using first-order logic formulas, in special employing Markov logic. Obviously, the representation of domain knowledge is delicate, given the flexibility and compactness of modeling of Markov logic. As an example, suppose one assumes the following formula as knowledge about a correct data partition:

$$\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Feature\_i}(\mathtt{x}) \wedge \mathtt{Feature\_j}(\mathtt{x}) \Rightarrow \mathtt{BelongsToCluster}(\mathtt{x}, \mathtt{Cluster\_1}),$$

when the correct model satisfies:

$$\forall x \forall y \, \texttt{Feature\_i}\,(x) \vee \texttt{Feature\_j}\,(x) \Rightarrow \texttt{BelongsToCluster}\,(x, \texttt{Cluster\_1}).$$

Syntatically, the difference between sentences is minimal (a disjunction is replaced by a conjuntion), but this can influence greatly the final configuration of clusters.

We now turn to semisupervised learning; in particular, to semisupervised learning with constraints. Again, we wish to use Markov logic to represent two kinds of constraints: must-links and cannot-links.

The use of a reduced amount of labeled and data extended by a large quantity of unlabeled instances together in semisupervised learning has generated a great deal of controversy, with evidence in favor of adding unlabeled data [19–21] and also evidence contrary to it [22, 23, 16, 24, 25]. It seems easy enough to add labeled data to a problem encoded in Markov logic. What is more difficult is to encode constraints such as must-links and cannot-links.

With MLNs, tasks involving semisupervised learning with constraints (SSLC) can be modeled in a simple way, given the availability of a proper syntax to define constraints. As a tutorial explanation of how this problem can be addressed, consider the example presented in Section 3.1. In that example, to represent must-links and cannot-links constraints over the class labels, we could create two additional predicates denoted by `MustLink(row_i,row_j)` and `CannotLink(row_i,row_j)`. The first means that two instanciations (realizations in tha dataset) of all atributes belong to the same class, and the latter, the opposite, i.e., they should not belong to the same class. To complete the model, we have also to specify other two logic formulas correlating the constraints and class predicates. These formulas are:

$$\texttt{MustLink}(\texttt{row\_i}, \texttt{row\_j}) \Rightarrow \texttt{Class}(\texttt{row\_i}, \texttt{value\_c}) \Leftrightarrow \texttt{Class}(\texttt{row\_j}, \texttt{value\_c})$$

$$\texttt{CannotLink}(\texttt{row\_i}, \texttt{row\_j}) \Rightarrow \texttt{Class}(\texttt{row\_i}, \texttt{value\_c}) \Leftrightarrow \texttt{Class}(\texttt{row\_j}, \texttt{value\_c}).$$

In order to propagate the effect of the contraints for all instanciations constrained, a transitivity relation could also be modeled by a logic formula, like in:

$$\texttt{MustLink}(\texttt{row\_i}, \texttt{row\_j}) \wedge \texttt{MustLink}(\texttt{row\_j}, \texttt{row\_k}) \Rightarrow \texttt{MustLink}(\texttt{row\_i}, \texttt{row\_k}).$$

Note that cannot-links are not transitive, thus a sentence imposing such a transitivity relation should not be added. Now we can extend the methodology previously discussed in connection with SL, by suitably inserting some extra steps to create schemes for SSLC using Markov logic. These steps are:

1. Define predicates to represent must and cannot-link constraints.
2. Build formulas correlating constraint predicates and associated class label predicates
3. Build a formula to represent the transitivity relation between must-link contraint predicates.

4 Conclusion

The results presented in this paper indicate that Markov logic is a flexible and useful language to build classifiers. Supervised classifiers based on Markov logic suggest that this language can sometimes exceed the performance of state-the-art methods on real data. Besides, the main advantage of Markov logic classifiers is the possibility of encoding deterministic and substantial relations about the domain of interest.

Our main contribution is a methodology on how to encode classification problems using Markov logic. Formulas that encode deterministic contraints in semisupervised learning with constraints can be represented through Markov logic networks. The goal of the work is to provide methods and tools that allow an easy manipulation of constraints with Markov logic.

As future work, we intend to run experiments with Markov logic in unsupervised and semisupervised settings, so as to verify the extent to which our modeling proposals apply in those tasks. Yet in supervised classification, we intend to refine our model of learning employing a technique for attribute selection. We will also investigate the learning of classification rules by means of an inductive logic programming system, in order to use them in Markov logic framework.

Acknowledgments. This work has been developed in collaboration with HP.

References

1. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (2006) 107–136
2. Koller, D., Friedman, N., Getoor, L., Taskar, B.: Graphical models in a nutshell. In Getoor, L., Taskar, B., eds.: An Introduction to Statistical Relational Learning. MIT Press (2007)
3. Kok, S., Singla, P., Richardson, M., Domingos, P.: The alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington (2005)
4. Devroye, L., Gyorfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Springer Verlag, New York (1996)
5. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31**(3) (1999) 264–323
6. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning. MIT Press, Cambridge, Massachusetts, USA (2006)
7. Law, M.H.C., Topchy, A.P., Jain, A.K.: Model-based clustering with probabilistic constraints. In: SDM. (2005)
8. Shental, N., Bar-Hillel, A., Hertz, T., Weinshall, D.: Computing Gaussian mixture models with EM using equivalence constraints. In: NIPS. (2003)
9. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Royal Statistical Society Series B (1977)

10. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2004)
11. Genesereth, M.R., Nilsson, N.J.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers (1987)
12. Singla, P., Domingos, P.: Memory-efficient inference in relational domains. In: AAAI, AAAI Press (2006)
13. Singla, P., Domingos, P.: Discriminative training of Markov logic networks. In: AAAI, AAAI Press / The MIT Press (2005)
14. Park, J.D.: Using weighted max-sat engines to solve mpe. In: Eighteenth national conference on Artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence (2002) 682–687
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann (October 1999)
16. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning **29**(2-3) (1997) 131–163
17. Cohen, I., Cozman, F., Sebe, N., Cirelo, M.C., Huang, T.: Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(12) (2004) 1553–1568
18. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer (2001)
19. Castelli, V.: The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition. PhD thesis, Stanford University (December 1994)
20. Castelli, V., Cover, T.M.: On the exponential value of labeled samples. Pattern Recognition Letters **16** (1995) 105–111
21. O'Neill, T.J.: Normal discrimination with unclassified observations. Journal of the American Statistical Association **73**(364) (1978) 821–826
22. Shahshahani, B., Landgrebe, D.: The Effect of Unlabeled Samples in Reducing the Small Sample Size Problem and Mitigating the Hughes Phenomenon. IEEE Transactions on Geoscience and Remote Sensing (1994)
23. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning **39** (2000) 103–144
24. Baluja, S.: Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. In: Neural and Information Processing Systems (NIPS). (1998)
25. Cozman, F.G., Cohen, I.: Unlabeled data can degrade classification performance of generative classifiers. In: Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, Pensacola, Florida (2002) 327–331