Maximum Margin Bayesian Networks

A Conjugate Gradient Based Approach

Franz Pernkopf, Michael Wohlmayr and Sebastian Tschiatschek
{pernkopf,michael.wohlmayr,tschiatschek}@tugraz.at
Signal Processing and Speech Communication Laboratory,
Graz University of Technology,
Austria

August 24, 2011

Abstract

This document briefly describes the usage of the provided MATLAB code for learning Maximum Margin Bayesian Networks (MMBNs) based on an example. For a description of the underlying algorithm consider [1].

Contents

1 Capabilities **1**

2 Interface **1**
- 2.1 Example Network . . . 2
- 2.2 Adjacency Matrix . . . 2
- 2.3 Vector of Node Sizes . . . 2
- 2.4 Initial Probabilities . . . 3
- 2.5 Iterations . . . 3
- 2.6 Tuning Parameters . . . 4
- 2.7 Maximum Margin Probabilities . . . 4

3 Example Code **4**

1 Capabilities

The provided MATLAB code can be used to discriminatively learn the parameters of fully observed Bayesian networks (BN) with fixed structure and discrete nodes using the maximum margin criterion. The probabilities in the BN are represented by conditional probability tables (CPTs) associated with the nodes of the network.

2 Interface

`learnCGMMParameters` is the main function provided for learning MMBNs. It takes the following input arguments, that are described in more detail in the next sections:

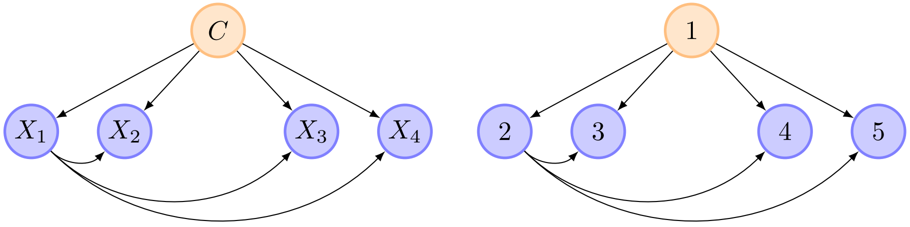
`A`	Adjacency matrix of the considered BN.
`ns`	Vector of node sizes.
`theta_initial`	Initial probabilities.
`samples`	Training samples.

`iterations`	Maximum number of conjugate gradient steps.
`kappa`	Tuning parameter.
`lambda`	Tuning parameter.

The function returns the vector `theta` containing the maximum margin optimized parameters of the network as the result.

2.1 Example Network

Throughout the explanation of the function `learnCGMMParameters` the BN presented in Fig. 1(a) is used as an example. The order of the nodes has to be fixed to use the function. The class node has to be node 1 by definition. In the following, the order shown in Fig. 1(b) is assumed. Further, it is assumed that there are 5 different classes and that the cardinality of the nodes X_1, X_2, X_3 and X_4 is $3, 4, 2$ and 6, respectively.



(a) Bayesian network (b) Bayesian network with ordered nodes

Figure 1: Example Bayesian Network.

2.2 Adjacency Matrix

The adjacency matrix A of a directed graph $\mathcal{G} = (V, E)$, where $V = (v_1, \ldots, v_N)$, is defined by

$$(a_{ij})_{\substack{i=1,\ldots,N\\ j=1,\ldots,N}} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

(We implicitly assumed that the considered graphs must not contain multiple edges between any pair of nodes.) The parameter `A` is the adjacency matrix of the considered BN. Hence, in the case of the example network

$$\texttt{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

2.3 Vector of Node Sizes

The vector `ns` specifies the number of different values every node can take, i.e. the cardinality. The order of the entries must correspond to the order of the nodes used for the adjacency matrix.

For the considered example:

$$\texttt{ns} = \begin{pmatrix} 5 & 3 & 4 & 2 & 6 \end{pmatrix}^T.$$

2.4 Initial Probabilities

`theta_initial` is a vector containing the initial probabilities of the BN, i.e. before maximum margin learning takes place. Typically, `theta_initial` are the maximum likelihood (ML) probabilities of the training samples. The ML parameters can be determined using the provided function `learnMLParameters(...)`.

The organization of the probabilities in the vector is as follows: The CPTs of the nodes in the BN are represented as vectors $\texttt{theta\_initial}_1, \ldots, \texttt{theta\_initial}_N$ and stacked upon each other in the order of the nodes, i.e.

$$\texttt{theta\_initial} = \left(\texttt{theta\_initial}_1^T \quad \ldots \quad \texttt{theta\_initial}_N^T\right)^T .$$

Representation of the Conditional Probability Table of node K_0 as the Vector $\texttt{theta\_initial}_{\texttt{K}_0}$. Consider the BN presented in Fig. 2. The node K_0 has the parent nodes $K_1, \ldots, K_S$. Further, it is assumed that $K_1 < K_2 < \ldots < K_S$ and that the nodes $K_0, \ldots, K_S$ can take $|K_0|, \ldots, |K_S|$ values, respectively. Then, the I^{th} entry of $\texttt{theta\_initial}_{\texttt{K}_0}$, where

$$I = k_0 + \sum_{j \in \{1,\ldots,S\}} \left[(k_j - 1)|K_0| \prod_{j<l\leq S} |K_l| \right], \tag{2}$$

corresponds to the probability $p(K_0 = k_0 | K_1 = k_1, \ldots, K_S = k_S)$.

The index I can also be determined using the MATLAB command `sub2ind`:

```
I = sub2ind(ns([K0, K1, ..., KS]), k0, k1, ..., ks)
```

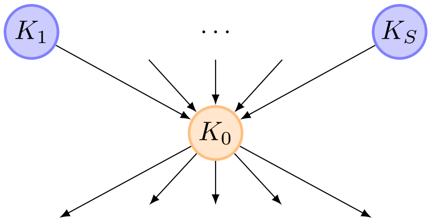


Figure 2: Node of a Bayesian network

Training Samples

The training samples must be supplied as a matrix with each column representing a fully observed training sample, i.e.

$$\texttt{samples} = \left[\mathbf{s}_1, \ldots, \mathbf{s}_M\right],$$

where $\mathbf{s}_i$ is the i^{th} training sample in form of a column vector. Each vector $\mathbf{s}_i$ has N components corresponding to observations of the nodes in the order used for the adjacency matrix. Remember that node 1 represents the class variable. The n^{th} entry of the i^{th} training sample must be an element in $\{1, \ldots, \texttt{ns}(n)\}$. If this does not hold naturally for your training data apply a suitable transformation (usually a simple shift of the values).

2.5 Iterations

The parameter `iterations` is the maximum number of conjugate gradient steps the method is allowed to perform.

2.6 Tuning Parameters

The tuning parameters `kappa` and `lambda` are scalars. Their meaning is described in [1].

2.7 Maximum Margin Probabilities

At the end of the training process the function returns the vector `theta` with the maximum margin optimized parameters. The entries of this vector are sorted as described in Section 2.4.

3 Example Code

An example demonstrating the provided code can be found in the file `run.m` and is presented in Listing 1.

Listing 1: Example usage of the provided MATLAB code (`run.m`).

```
% Maximum Margin Bayesian Network Classifiers example script
%
% The provided software may be used free of charge for research purposes.
% For other uses and further support please contact Franz Pernkopf,
% pernkopf@tugraz.at.
%
% References:
%  F. Pernkopf, M. Wohlmayr, and S. Tschiatschek,
%  "Maximum Margin Bayesian Network Classifiers,"
%  IEEE Transactions on Pattern Analysis and Machine Intelligence,
%  vol. 99, no. PrePrints, 2011
%
% Authors:
%  F. Pernkopf, M. Wohlmayr, and S. Tschiatschek
%
% Version:
%  1.0 August 24th, 2011
%

%% (*) clear environment
clear all
clc

%% Values to use to rediscover Table 1 and Table 2 from
%% TODO: Insert publication name

% MNIST
%               NB       TAN-CMI        TAN-OMI-CR     TAN-CR
% kappa         0.025    0.075          0.25           0.05
% lambda        0.2      0.1            0.075          0.1
% iter (MM)     47       39             33             34
% iter (CLL)    43       17             24             21

% USPS
%               NB       TAN-CMI        TAN-OMI-CR     TAN-CR
% kappa         0.075    0.05           0.01           0.075
% lambda        0.05     0.07           0.06           0.05
% iter (MM)     29       28             17             21
% iter (CLL)    33       53             30             43

%% (0) Specify simulation

% dataset to use
dataset = 'USPS';
%dataset = 'MNIST';

% network structure to use
structure = 'NB';
%structure = 'TAN-CMI';
%structure = 'TAN-OMI-CR';
%structure = 'TAN-CR';

kappa = 0.075;
lambda = 0.05;
iter = 29;
iterCLL = 33;
```

```
% smoothing parameter
epsilon = 1e-5;

%% (1) load training and test data
% training data: USPS_Train.mat
% test data: USPS_Test.mat
% both contain matrices with samples that are organized as follows:
%   each column contains one training samples
%   each row is a specific attribute, where the first-row is the class
%     variable

% load training set
load(['datasets/', dataset, '_Train', '.mat']);
TrainData = Samples;

% load test set
load(['datasets/', dataset, '_Test', '.mat']);
TestData = Samples;

%% (2) Specify structure of graphical network by an adjacency matrix and
%% the number of values each attribute can take

struct_file = ['structures/', dataset, '_', structure, '.mat'];

% load adjecency matrix
load(struct_file);
nrNodes = size(A, 1);

% set up node sizes ns
ns = max(TrainData, [], 2);

%% dump problem information
nrCPTentries = 0;
for n = 1:nrNodes,
    nrParentStates = prod(ns(logical(A(:,n))));
    nrCPTentries = nrCPTentries + nrParentStates*ns(n);
end

fprintf(1, '>> Problem information\n');
fprintf(1, 'Training Data: %d samples\n', size(TrainData, 2));
fprintf(1, 'Test Data: %d samples\n', size(TestData, 2));
fprintf(1, 'Number of CPT-entries: %d entries\n\n', nrCPTentries);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ML learning %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% (3) Calculate ML parameters from training data
fprintf(1, '>> Learning ML parameters...\n');
tic;
thetaML = learnMLParameters(A, ns, TrainData);
% Smooth probabilities and renormalize
thetaML(thetaML == 0) = epsilon;
thetaML = normalize(A, ns, thetaML);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);

%% (4) classify data using the obtained parameters
fprintf(1, '>> Classification using ML parameters...\n');
tic;
[CLRateTrain, CLTableTrain] = ClassifyData(A, ns, thetaML, TrainData);
[CLRateTest, CLTableTest] = ClassifyData(A, ns, thetaML, TestData);
fprintf(1, '* Training set: %3.2f %%\n', CLRateTrain);
fprintf(1, '* Test set: %3.2f %%\n', CLRateTest);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CLL learning %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% (3) Train the network
tic;
fprintf(1, '>> Learning CLL parameters...\n');
thetaCLL = learnCGCLLParameters(A, ns, thetaML, TrainData, iterCLL);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);
```

```
%% (4) classify data using the obtained parameters
fprintf(1, '>> Classification using CLL parameters...\n');
tic;
[CLRateTrain, CLTableTrain] = ClassifyData(A, ns, thetaCLL, TrainData);
[CLRateTest, CLTableTest] = ClassifyData(A, ns, thetaCLL, TestData);
fprintf(1, '* Training set: %3.2f %%\n', CLRateTrain);
fprintf(1, '* Test set: %3.2f %%\n', CLRateTest);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MM learning %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% (3) Train the network
tic;
fprintf(1, '>> Learning MM parameters...\n');
thetaMM = learnCGMMParameters(A, ns, thetaML, TrainData, iter, kappa, lambda);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);

%% (4) classify data using the obtained parameters
fprintf(1, '>> Classification using MM parameters...\n');
tic;
[CLRateTrain, CLTableTrain] = ClassifyData(A, ns, thetaMM, TrainData);
[CLRateTest, CLTableTest] = ClassifyData(A, ns, thetaMM, TestData);
fprintf(1, '* Training set: %3.2f %%\n', CLRateTrain);
fprintf(1, '* Test set: %3.2f %%\n', CLRateTest);
tp = toc;
fprintf(1, '<< (took %e s)\n\n', tp);
```

References

[1] F. Pernkopf, M. Wohlmayr, and S. Tschiatschek, "Maximum margin bayesian network classifiers," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 99, no. PrePrints, 2011.