OOP project – 2014/15

Learning dynamic Bayesian networks

March 30, 2015

Motivation

- **Multivariate time series** (MTS) arise in several interesting contexts and provide an opportunity for studying changes over time.
- **Dynamic Bayesian networks** (DBN) are a probabilistic graphical models for describing multivariate temporal processes.
- DBN are able to express two different relations:
  1. the **intra-temporal** relations, expressing the dependences between the variables at a particular time instant t, and
  2. the **inter-temporal** relations, specifying how the variables observed at times $0, \ldots, t$ affect the variables at $t + 1$.
- DBNs are defined upon Bayesian networks (BN). BN are used to model **time-invariant** processes, whereas DBN model **time-variant** ones.

Table of Contents

Bayesian network

A **Bayesian network** (BN) is a triple $B = (\vec{X}, G, \Theta)$:

1. $\vec{X} = (X_1, \ldots, X_n)$ is a random vector.
2. $G = (\vec{X}, E)$ is a **directed acyclic graph** (DAG) whose edges in E specify conditional dependencies between the variables.
3. $\Theta = \{\theta_{ijk}\}_{ijk}$ is the family of **network parameters**, defining the conditional probabilities.

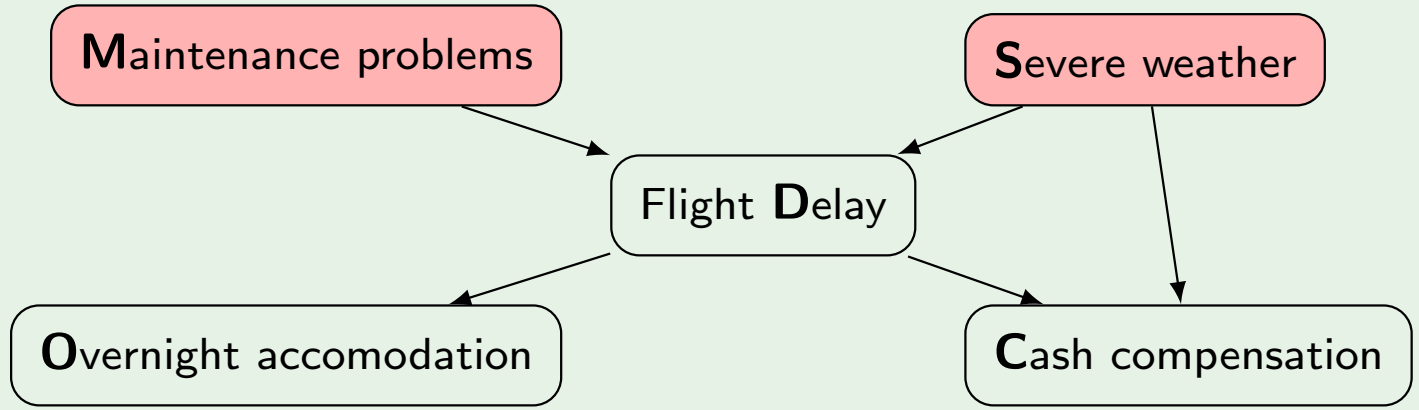
A BN defines a **joint probability distribution over** $\vec{X}$ given by

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(X_i \mid \vec{pa}(X_i)),$$

where $\vec{pa}(X_i)$ is the **set of parents of** X_i **in** G.

Bayesian networks

Simple example regarding airline regulations



$P(\mathbf{M} = \mathtt{T}) = 0.02$

$P(\mathbf{S} = \mathtt{T}) = 0.03$

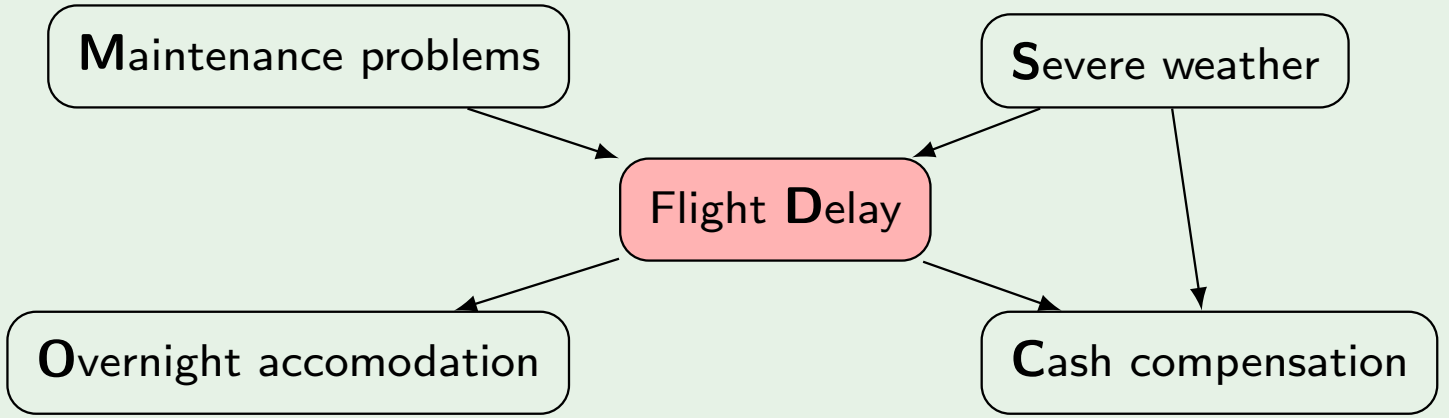
D	$P(\mathbf{O} = \mathtt{T} \mid \mathbf{F})$
T	0.3
F	0.01

M	S	$P(\mathbf{D} = \mathtt{T} \mid \mathbf{M}, \mathbf{S})$
T	T	0.95
T	F	0.2
F	T	0.6
F	F	0.01

D	S	$P(\mathbf{C} = \mathtt{T} \mid \mathbf{D}, \mathbf{S})$
T	T	0.05
T	F	0.7
F	T	0.01
F	F	0.02

Bayesian networks

Simple example regarding airline regulations



$P(\mathbf{M} = \mathtt{T}) = 0.02$

$P(\mathbf{S} = \mathtt{T}) = 0.03$

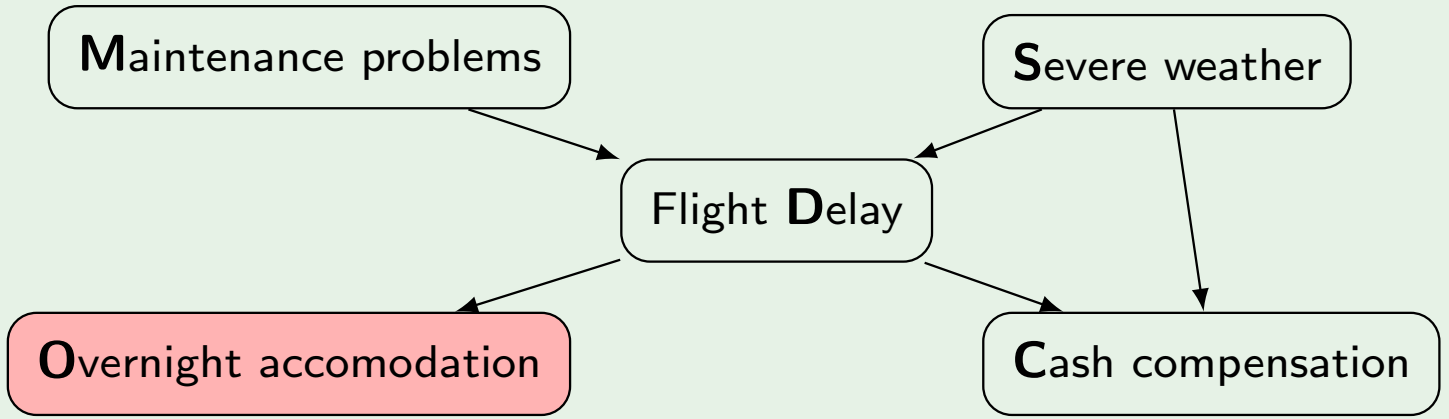
M	S	$P(\mathbf{D} = \mathtt{T} \mid \mathbf{M}, \mathbf{S})$
T	T	0.95
T	F	0.2
F	T	0.6
F	F	0.01

D	$P(\mathbf{O} = \mathtt{T} \mid \mathbf{F})$
T	0.3
F	0.01

D	S	$P(\mathbf{C} = \mathtt{T} \mid \mathbf{D}, \mathbf{S})$
T	T	0.05
T	F	0.7
F	T	0.01
F	F	0.02

Bayesian networks

Simple example regarding airline regulations



$P(\mathbf{M} = \mathtt{T}) = 0.02$

$P(\mathbf{S} = \mathtt{T}) = 0.03$

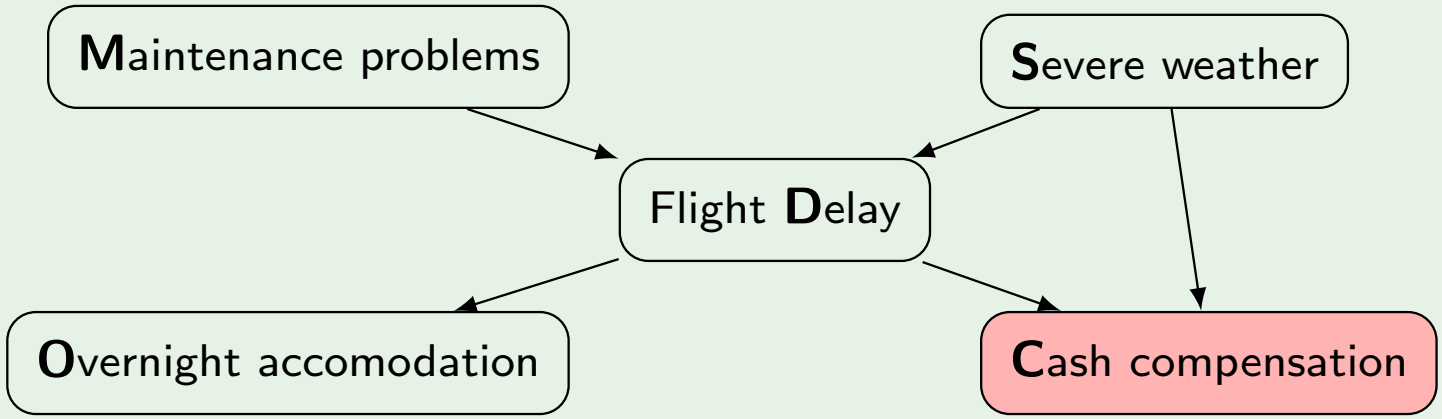
D	$P(\mathbf{O} = \mathtt{T}\|\mathbf{F})$
T	0.3
F	0.01

M	S	$P(\mathbf{D} = \mathtt{T}\|\mathbf{M}, \mathbf{S})$
T	T	0.95
T	F	0.2
F	T	0.6
F	F	0.01

D	S	$P(\mathbf{C} = \mathtt{T}\|\mathbf{D}, \mathbf{S})$
T	T	0.05
T	F	0.7
F	T	0.01
F	F	0.02

Bayesian networks

Simple example regarding airline regulations



$P(\mathbf{M} = \mathtt{T}) = 0.02$

$P(\mathbf{S} = \mathtt{T}) = 0.03$

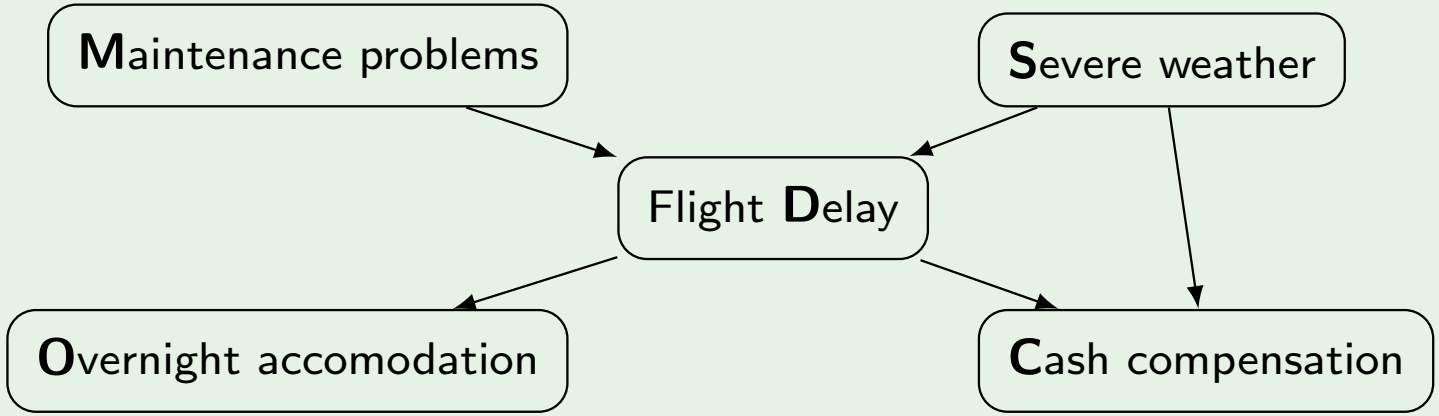
M	S	$P(\mathbf{D} = \mathtt{T} \mid \mathbf{M}, \mathbf{S})$
T	T	0.95
T	F	0.2
F	T	0.6
F	F	0.01

D	$P(\mathbf{O} = \mathtt{T} \mid \mathbf{F})$
T	0.3
F	0.01

D	S	$P(\mathbf{C} = \mathtt{T} \mid \mathbf{D}, \mathbf{S})$
T	T	0.05
T	F	0.7
F	T	0.01
F	F	0.02

Bayesian networks

Simple example regarding airline regulations



- Factorized joint probability distribution:

$$P(M, S, D, O, C) = P(M)\ P(S)\ P(D|M, S)\ P(O|D)\ P(C|D, S).$$

- Network has 12 parameters, while complete specification would require $2^5 - 1 = 31$ conditional probabilities (61% space savings).

Learning Bayesian networks

Given a dataset D comprising instances of $\vec{X}$, find the structure G and parameters Θ that best fits the data.

- Define a **scoring criterion** $\phi(B, D)$ to measure network fitness.
- Define a **search procedure** to generate networks whose score is evaluated.

This is called **score-based learning**.

Learning Bayesian networks

Score-based learning can be extremely efficient if the scoring criterion employed is decomposable:

- In that case it is named **local score-based learning**.
- The score of the network being considered can be updated locally as a result of local network changes of previously evaluated networks.

A **decomposable scoring criterion** can be expressed as a sum of local terms depending on a node and its parents:

$$\phi(B, D) = \sum_{i=1}^{n} \phi_i(\vec{pa}(X_i), D).$$

Learning Bayesian networks

Commonly used decomposable scores:

- **Log-likelihood** (LL) tends to favour complete network structures, which typically leads to over-fitting:

$$\mathrm{LL}(B|D) = -\log P_B(D)$$

- **Minimum description length** (MDL) extends LL by including a term to penalize complex network structures:

$$\mathrm{MDL}(B|D) = \mathrm{LL}(B|D) - \frac{1}{2}\log(N)|B|,$$

where $|B|$ is the number of parameters of network B.

Hardness results

- In general, learning unrestricted BN is NP-hard.
- The standard methodology for learning BN became **heuristic search**, based on scoring metrics optimization.
- The **Greedy Hill Climbing (GHC)** is a well known heuristic algorithm to learn BN.

Greedy Hill Climbing

- Suppose we have some network structure, which is a DAG. We can define its **neighbourhood** in **DAG-space** as all networks we can reach by applying one of the **operators**:
  - **add** an edge;
  - **delete** an edge;
  - **reverse** an edge.
- At each search step, we find the best neighbour and move to it.
- But, it must be the best neighbour in DAG-space!
- When removing an edge from a DAG the resulting network will always be a DAG, but when adding or flipping an edge from a DAG we need to guarantee that the resulting network is still a DAG.

Greedy Hill Climbing

GHC algorithm for learning BN

Input: initial structure $\mathcal{N}_{init}$, dataset D, a scoring function ϕ, stopping criteria $\mathcal{C}$

Output: final structure $\mathcal{N}_{res}$

1. $\mathcal{N}_{res} = \mathcal{N}_{init}$ and $\mathcal{N}' = \mathcal{N}_{res}$
2. while $\mathcal{C}$ is not satisfied
3. $\quad \mathcal{N}'' = \arg\max_{\mathcal{N} \in \text{neighbourhood}(\mathcal{N}')} \phi(\mathcal{N})$
4. $\quad$ if $\phi(\mathcal{N}'') > \phi(\mathcal{N}_{res})$ then $\mathcal{N}_{res} = \mathcal{N}''$
5. $\quad \mathcal{N}' = \mathcal{N}''$
6. end while
7. return $\mathcal{N}_{res}$

Greedy Hill Climbing

GHC can get stuck in local optima. Two simple strategies can be effective to escape from these local optima:

- **TABU list**: do not revisit recently seen structures;
- **Random restarts**: apply some operators at random when at a local optimum.

Greedy Hill Climbing

GHC algorithm for learning BNs, with TABU list and random restarts

Input: initial structure $\mathcal{N}_{init}$, dataset D, a scoring function ϕ, stopping criteria $\mathcal{C}$

Output: final structure $\mathcal{N}_{res}$

1. $\mathcal{N}_{res} = \mathcal{N}_{init}$, $\mathcal{N}' = \mathcal{N}_{res}$ and TABU $= \{\mathcal{N}_{res}\}$
2. while $\mathcal{C}$ is not satisfied
3. $\quad \mathcal{N}'' = \arg\max_{\mathcal{N} \in \text{neighbourhood}(\mathcal{N}') \text{ and } \mathcal{N} \notin \text{TABU}} \phi(\mathcal{N})$
4. $\quad$ if $\phi(\mathcal{N}') > \phi(\mathcal{N}'')$ then $\mathcal{N}'' = \text{random}(\mathcal{N}')$
5. $\quad$ if $\phi(\mathcal{N}'') > \phi(\mathcal{N}_{res})$ then $\mathcal{N}_{res} = \mathcal{N}''$
6. $\quad$ TABU $=$ TABU $\cup \mathcal{N}'$
7. $\quad \mathcal{N}' = \mathcal{N}''$
8. end while
9. return $\mathcal{N}_{res}$

Greedy Hill Climbing

The stopping criterion $\mathcal{C}$ may vary, for instance:

- the score of the best network exceeds a threshold;
- a local maximum is attained;
- maximum number of random restarts.

Typically the number of parents a node can have is also upper bounded!

Table of Contents

Dynamic Bayesian networks

Dynamic Bayesian networks (DBN) extend BN to model temporal processes.

A DBN is composed by:

- a **prior network** B_0, which specifies a joint probability distribution over $\vec{X}[0]$.
- a **set of transition networks** B_0^t over the variables $\vec{X}[0], \ldots, \vec{X}[t]$, specifying the state transition probabilities, for $0 < t \leq T$.

Dynamic Bayesian networks

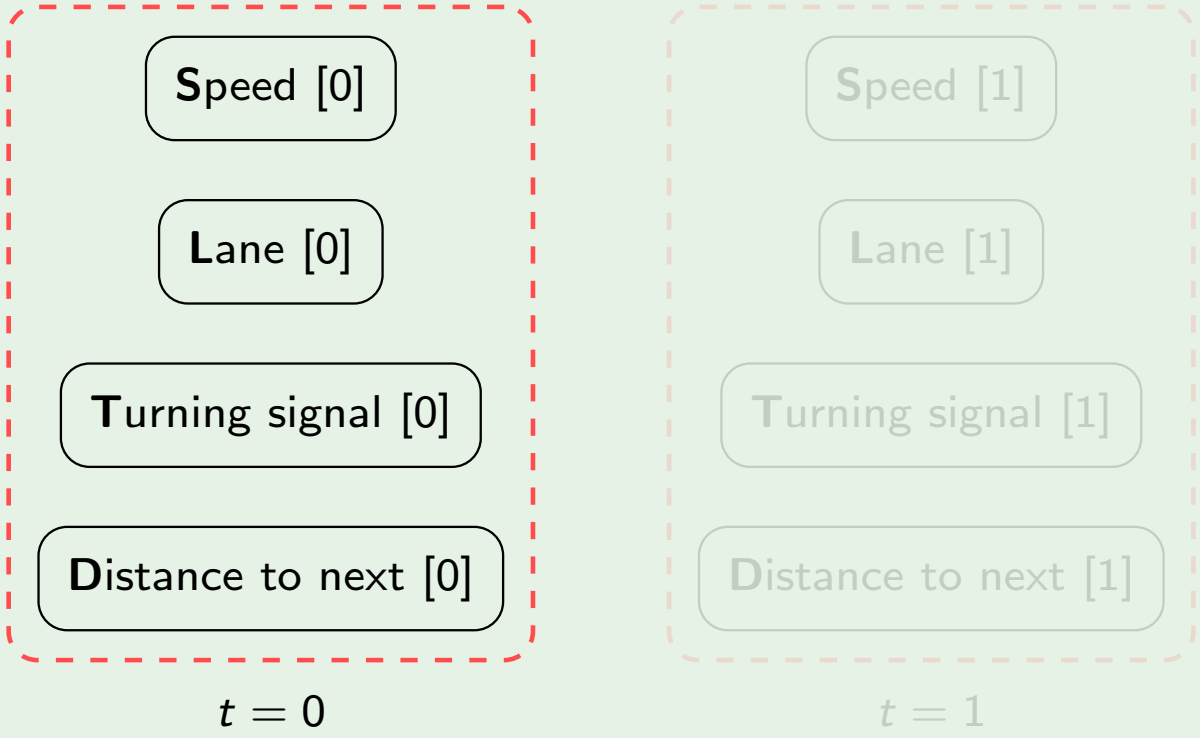
Usual simplifying assumptions:

- **first-order Markov**: $P(\vec{X}[t+1] \mid \vec{X}[0], ..., \vec{X}[t]) = P(\vec{X}[t+1] \mid \vec{X}[t])$.
- **stationary**: transition probabilities independent of t.

When the DBN is first-order Markov and stationary, also called **homogeneous**, the conditional probability $P(\vec{X}[t+1] \mid \vec{X}[t])$ is the same for all t.

Dynamic Bayesian networks

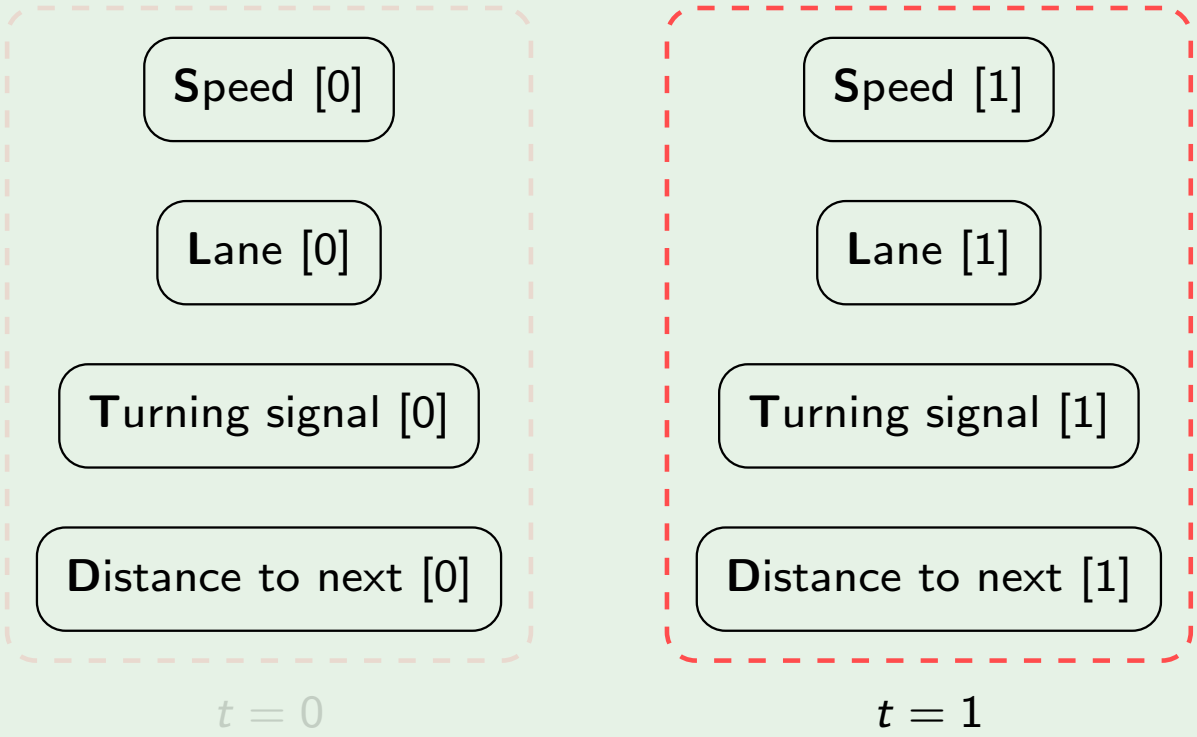
Simple example regarding driver behaviour



Time-slices

Dynamic Bayesian networks

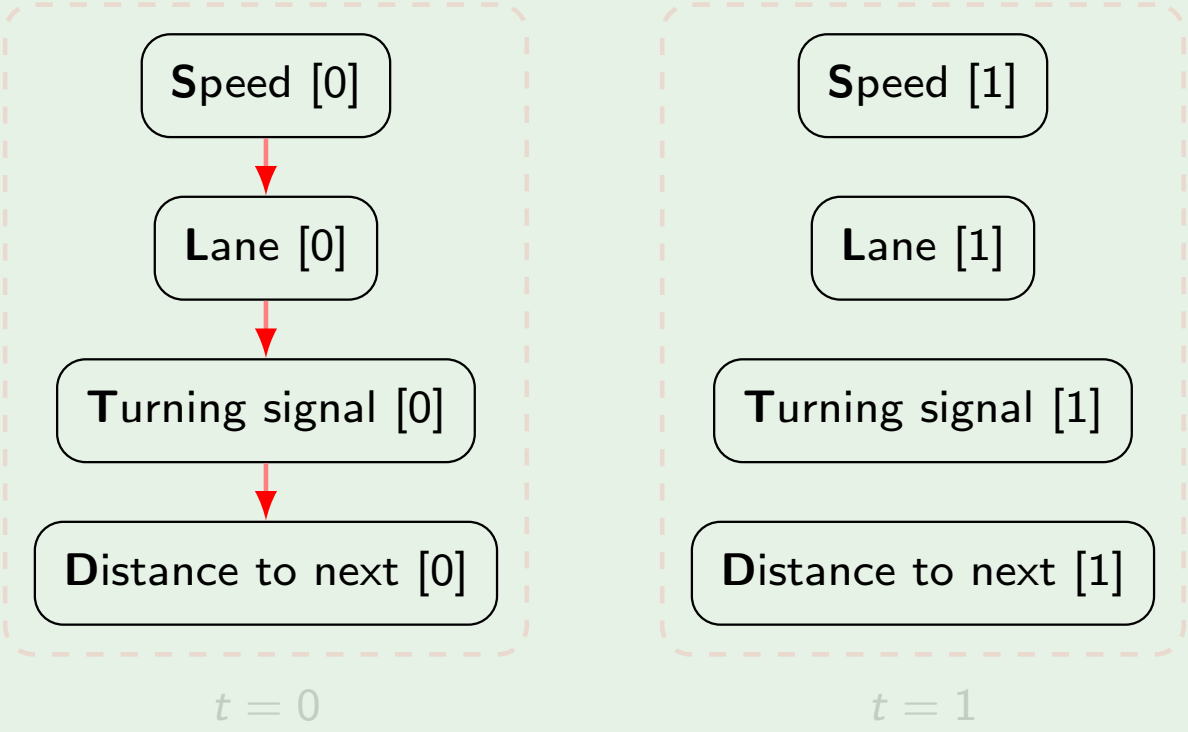
Simple example regarding driver behaviour



Time-slices

Dynamic Bayesian networks

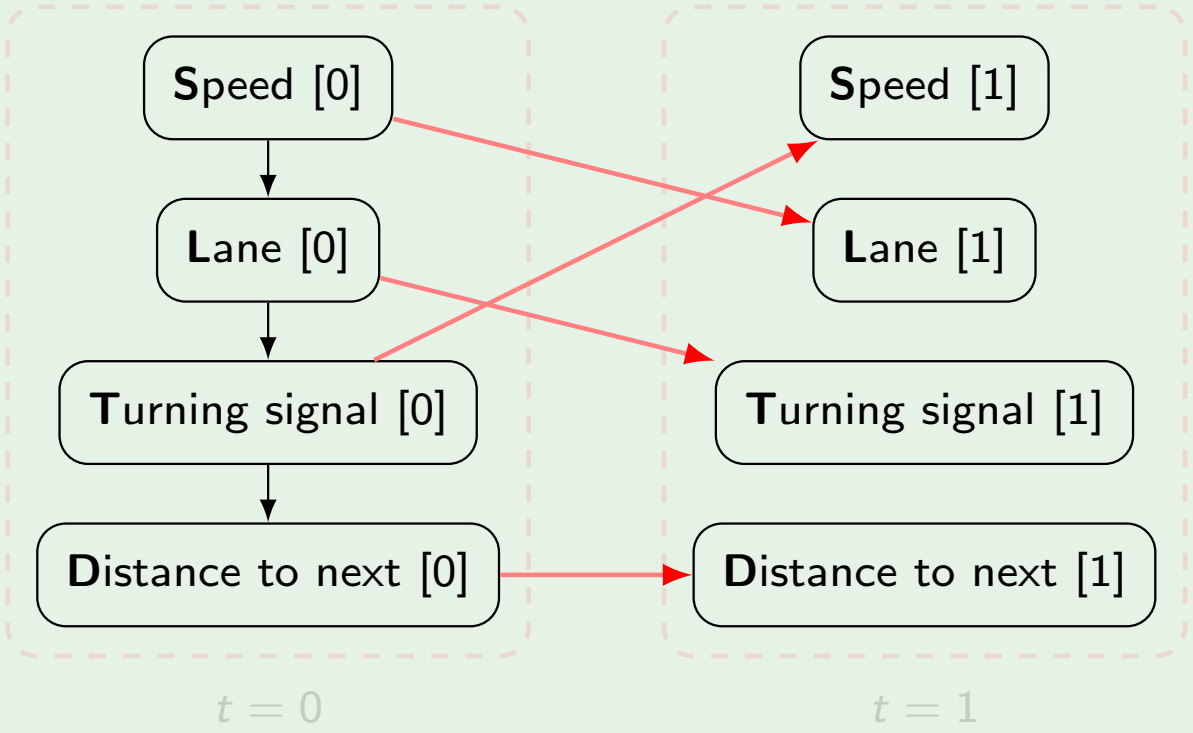
Simple example regarding driver behaviour



Prior relations

Dynamic Bayesian networks

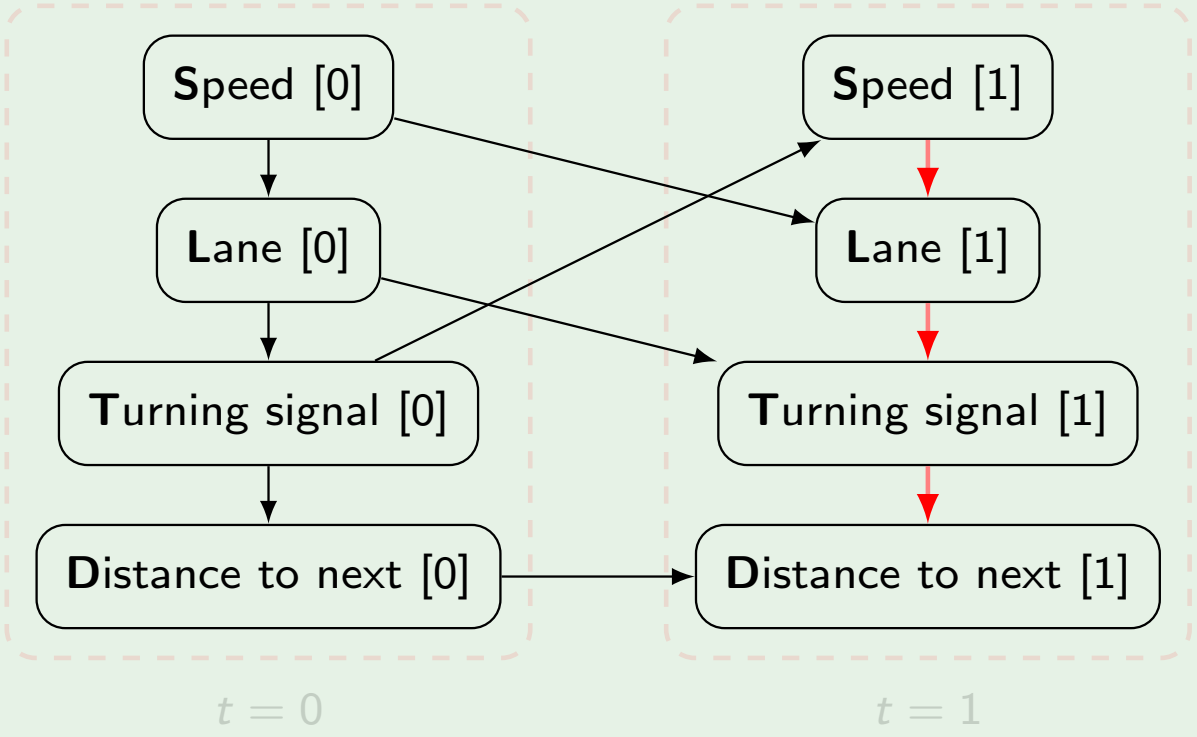
Simple example regarding driver behaviour



Inter-slice relations

Dynamic Bayesian networks

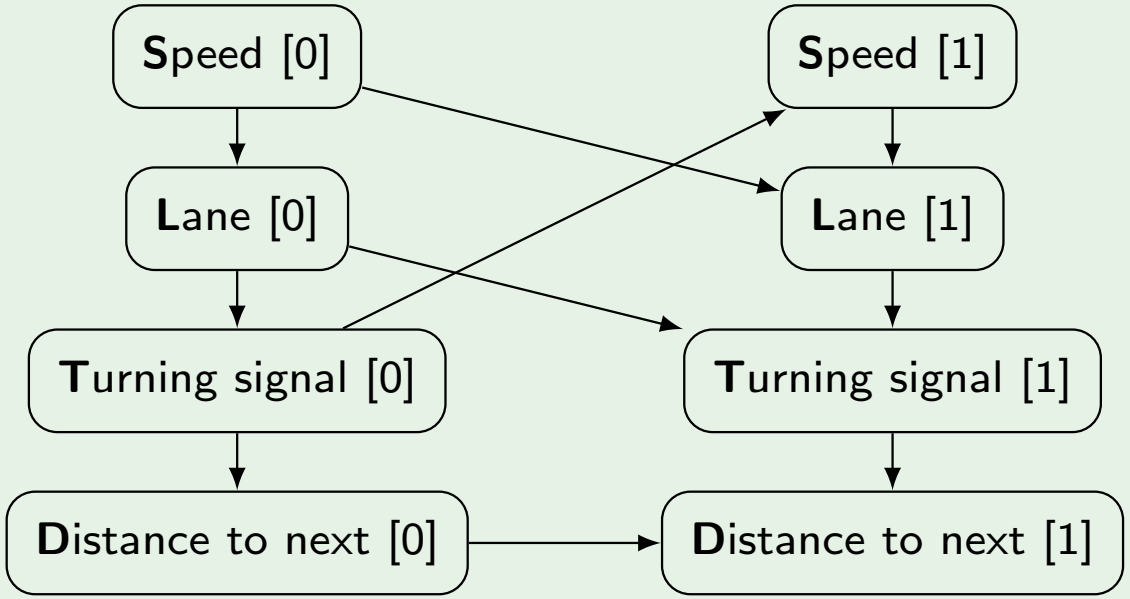
Simple example regarding driver behaviour



Intra-slice relations

Dynamic Bayesian networks

Simple example regarding driver behaviour



Multivariate time-series dataset:

S[0]	L[0]	T[0]	D[0]	S[1]	L[1]	T[1]	D[1]
20	0	left	50	30	0	none	80
100	2	none	500	120	2	none	300
...	...	...	...	...	...	...	...

Learning dynamic Bayesian networks

Typical data to learn DBN:

$X_1[0]$,	$\ldots$,	$X_n[0]$,	$X_1[1]$,	$\ldots$,	$X_n[1]$,	$\ldots$,	$X_1[T]$,	$\ldots$,	$X_n[T]$
$v_{11}[0]$,	$\ldots$,	$v_{n1}[0]$,	$v_{11}[1]$,	$\ldots$,	$v_{n1}[1]$,	$\ldots$,	$v_{11}[T]$,	$\ldots$,	$v_{n1}[T]$
$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$
$v_{1N}[0]$,	$\ldots$,	$v_{nN}[0]$,	$v_{1N}[1]$,	$\ldots$,	$v_{nN}[1]$,	$\ldots$,	$v_{1N}[T]$,	$\ldots$,	$v_{nN}[T]$
t=0			t=1				t=T		

However, all instances in the data need not to have the same size, and so the data might not be rectangular.

Learning dynamic Bayesian networks

Typical data to learn homogenous DBN:

All pairs (t,t+1):

$X_1[t]$,	$\ldots$,	$X_n[t]$,	$X_1[t+1]$,	$\ldots$,	$X_n[t+1]$
(0,1)					
$v_{11}[0]$,	$\ldots$,	$v_{n1}[0]$,	$v_{11}[1]$,	$\ldots$,	$v_{n1}[1]$
$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$
$v_{1N}[0]$,	$\ldots$,	$v_{nN}[0]$,	$v_{1N}[1]$,	$\ldots$,	$v_{nN}[1]$
(1,2)					
$v_{11}[1]$,	$\ldots$,	$v_{n1}[1]$,	$v_{11}[2]$,	$\ldots$,	$v_{n1}[2]$
$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$
$v_{1N}[1]$,	$\ldots$,	$v_{nN}[1]$,	$v_{1N}[2]$,	$\ldots$,	$v_{nN}[2]$
$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$
(T–1,T)					
$v_{11}[T-1]$,	$\ldots$,	$v_{n1}[T-1]$,	$v_{11}[T]$,	$\ldots$,	$v_{n1}[T]$
$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$,	$\ldots$
$v_{1N}[T-1]$,	$\ldots$,	$v_{nN}[T-1]$,	$v_{1N}[T]$,	$\ldots$,	$v_{nN}[T]$

Again, data might not be rectangular!

Learning dynamic Bayesian networks

Learning an homogeneous DBN corresponds to learn two networks:

- the initial network B_0:
  - learned from $\vec{X}[0]$ only;
  - learned with BN version of GHC.
- the transition network B_t^{t+1}:
  - learned from all pairs $(\vec{X}[t], \vec{X}[t+1])$ with $0 \leq t < T$;
  - need to adapt BN version of GHC to consider the restriction that edges may appear from time-slice t to time-slice $t+1$ (inter-slice relations), or from time-slice $t+1$ to $t+1$ (intra-slice relations), only;
  - DAG restriction need only to be verified when considering intra-slice relations;
  - the local contribution of node $X_i[t+1]$ to the global score of B_t^{t+1} is given by the parents from the same time-slice $t+1$ as well as the parents from the previous time-slice t.

Inference with dynamic Bayesian networks

After learning a DBN, we typically want to know the value of some variable $X_i[t+1]$ given that we already know the values of $\vec{X}[t]$.

To this end we need to compute for every possible value x_{ik} of $X_i[t+1]$ the one that yields the highest probability given by:

$$P\left(X_i[t+1] = x_{ik} \mid \vec{X}[t] = (v_1, \ldots, v_n)\right) = \sum_{d_1=1}^{r_1} \cdots \sum_{d_{i-1}=1}^{r_{i-1}} \sum_{d_{i+1}=1}^{r_{i+1}} \sum_{d_n=1}^{r_n}$$

$$P\left(\vec{X}[t+1] = (d_1, \ldots, d_{i-1}, x_{ik}, d_{i+1}, \ldots, d_n) \mid \vec{X}[t] = (v_1, \ldots, v_n)\right)$$