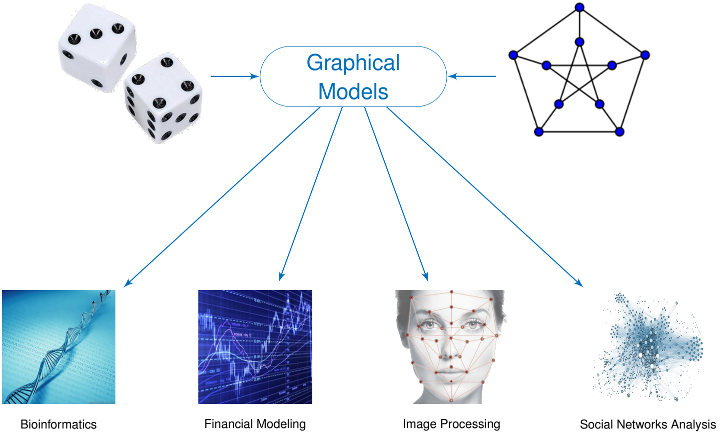
# Online Learning of Probabilistic Graphical Models

Frédéric Koriche
CRIL - CNRS UMR 8188, Univ. Artois
koriche@cril.fr

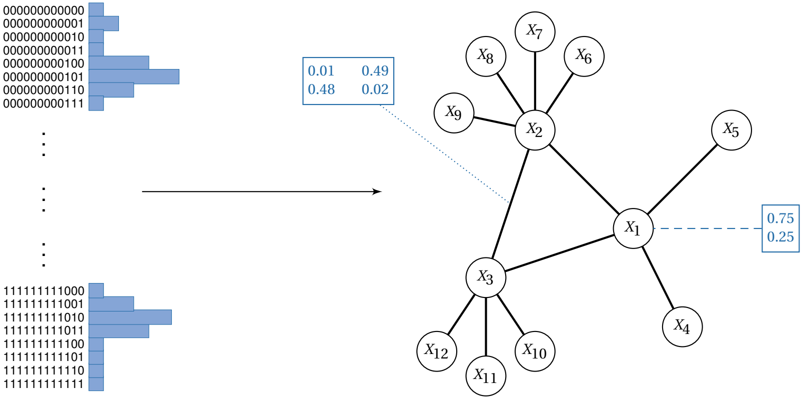
CRIL-U Nankin 2016

# Outline




## Graphical Models

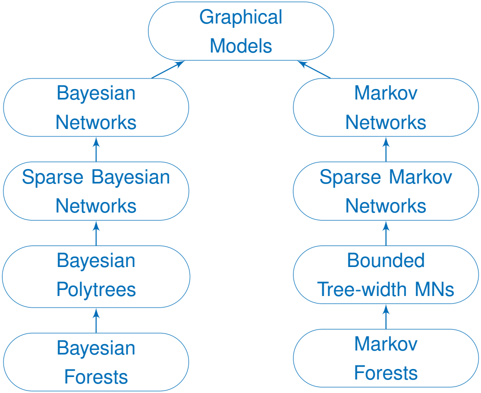
At the intersection of Probability Theory and Graph Theory, graphical models are a well-studied representation framework with various applications.




## Graphical Models

Encoding high-dimensional distributions in a compact and intuitive way:

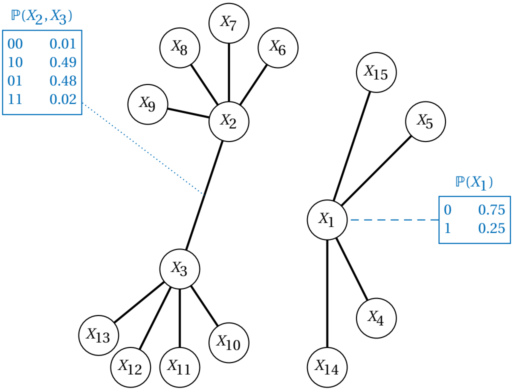
- Qualitative uncertainty (interdependencies) is captured by the structure
- Quantitative uncertainty (probabilities) is captured by the parameters




## Classes of Graphical Models

For an outcome space $\mathscr{X} \subseteq \mathbb{R}^n$, a class of graphical models is a pair $\mathscr{M} = \boldsymbol{G} \times \boldsymbol{\Theta}$, where $\boldsymbol{G}$ is space of $n$-dimensional graphs, and $\boldsymbol{\Theta}$ is a space of $d$-dimensional vectors.

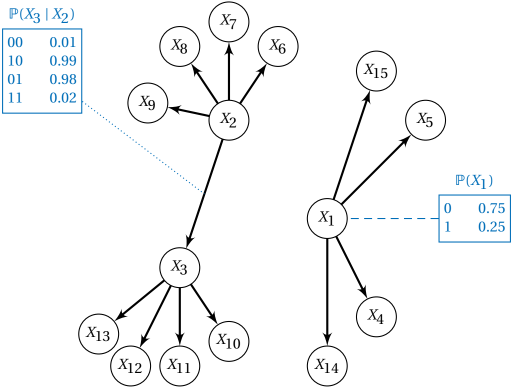
- $\boldsymbol{G}$ captures structural constraints (directed vs. undirected, sparse vs. dense, etc.)
- $\boldsymbol{\Theta}$ captures parametric constraints (binomial, multinomial, Gaussian, etc.)




## (Multinomial) Markov Forests

Using $[m] = \{0, \cdots, m-1\}$, the class of Markov Forests over $[m]^n$ is given by $\boldsymbol{F}_n \times \boldsymbol{\Theta}_{m,n}$, where

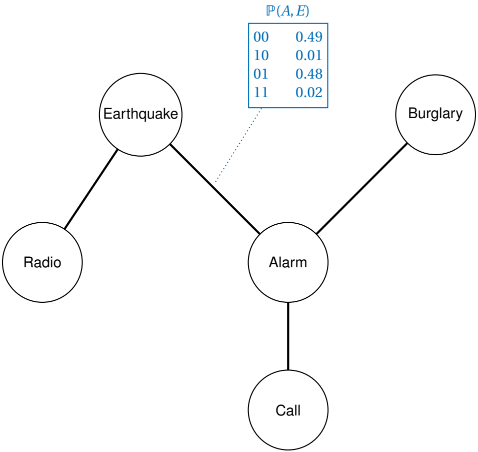
- $\boldsymbol{F}_n$ is the space of all acyclic graphs of order $n$;
- $\boldsymbol{\Theta}_{m,n}$ is the space of all parameter vectors mapping
  - a probability table $\boldsymbol{\theta}_i \subseteq [0,1]^m$ to each candidate node $i$, and
  - a probability table $\boldsymbol{\theta}_{ij} \subseteq [0,1]^{m \times m}$ to each candidate edge $(i,j)$.




## (Multinomial) Bayesian Forests

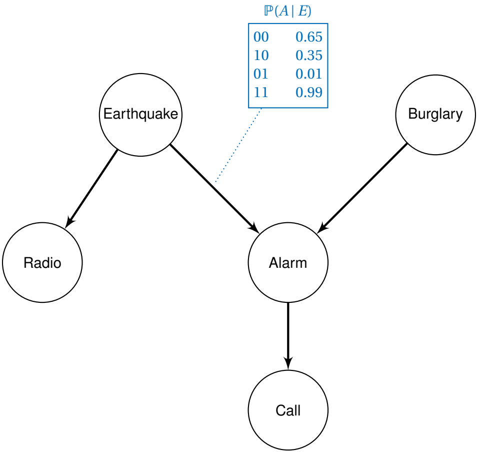
The class of Bayesian Forests over $[m]^n$ is given by $\overline{\boldsymbol{F}}_n \times \boldsymbol{\Theta}_{m,n}$, where

- $\overline{\boldsymbol{F}}_n$ is the space of all directed forests of order $n$;
- $\boldsymbol{\Theta}_{m,n}$ is the space of all parameter vectors mapping
  - a probability table $\boldsymbol{\theta}_i \subseteq [0,1]^m$ to each candidate node $i$, and
  - a conditional probability table $\boldsymbol{\theta}_{j|i} \subseteq [m] \times [0,1]^m$ to each candidate arc $(i,j)$.




## Example: The Alarm Model

- Markov tree representation
- Bayesian tree representation




## Example: The Alarm Model

- Markov tree representation
- Bayesian tree representation

# Outline

| $E$ | $B$ | $R$ | $A$ | $C$ |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| | | $\vdots$ | | |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |




## Learning

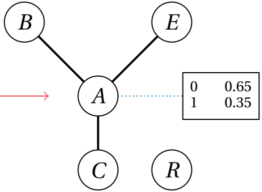
Given a class of graphical models $\mathscr{M} = \boldsymbol{G} \times \boldsymbol{\Theta}$, the learning problem is to extract from a sequence of outcomes $\boldsymbol{x}^{1:T} = (\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T)$,

- the structure $G \in \boldsymbol{G}$, and
- the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

of a generative model $M = (G, \boldsymbol{\theta})$ capable of predicting future, unseen, outcomes.
A natural loss function for measuring the performance of $M$ is the log-loss:

$$\ell(M, \boldsymbol{x}) = -\ln \mathbb{P}_M(\boldsymbol{x})$$

| $E$ | $B$ | $R$ | $A$ | $C$ |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| | | $\vdots$ | | |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |




## Learning

Given a class of graphical models $\mathscr{M} = \mathbf{G} \times \mathbf{\Theta}$, the learning problem is to extract from a sequence of outcomes $\boldsymbol{x}^{1:T} = (\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T)$,

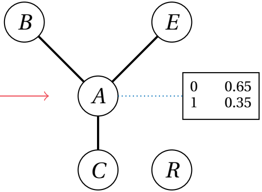
- the structure $G \in \mathbf{G}$, and
- the parameters $\boldsymbol{\theta} \in \mathbf{\Theta}$

of a generative model $M = (G, \boldsymbol{\theta})$ capable of predicting future, unseen, outcomes.

A natural loss function for measuring the performance of $M$ is the log-loss:

$$\ell(M, \boldsymbol{x}) = -\ln \mathbb{P}_M(\boldsymbol{x})$$

| $E$ | $B$ | $R$ | $A$ | $C$ |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| | | ⋮ | | |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |




## Learning

Given a class of graphical models $\mathscr{M} = \boldsymbol{G} \times \boldsymbol{\Theta}$, the learning problem is to extract from a sequence of outcomes $\boldsymbol{x}^{1:T} = (\boldsymbol{x}^1, \cdots, \boldsymbol{x}^T)$,

- the structure $G \in \boldsymbol{G}$, and
- the parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$

of a generative model $M = (G, \boldsymbol{\theta})$ capable of predicting future, unseen, outcomes.
A natural loss function for measuring the performance of $M$ is the log-loss:

$$\ell(M, \boldsymbol{x}) = -\ln \mathbb{P}_M(\boldsymbol{x})$$

000000010000
000000010001
000000000010
000000000011
010000000101
010000000101
100000000111
100000000111

⋮

110111111000
110111111001
111110111010
110111111011
111011111101
111011111101
111011111101
111111110111

Training Set

$$M = (G, \boldsymbol{\theta})$$

Graphical Model

## Batch Learning

A *two-stage* process:

- A model $M \in \mathscr{M}$ is first extracted from a training set.
- The average loss of the model $M$ is evalued using a test set.

$$M = (G, \boldsymbol{\theta}) \xrightarrow{\quad \frac{1}{T}\sum_{t=1}^{T} \ell(M, \boldsymbol{x}^t) \quad}$$

Graphical Model

000000010000
000000010001
000000000010
000000000011
010000000101
010000000101
100000000111

$\vdots$

110111111000
110111111001
111110111010
111011111101
111011111101
111011111101
111111110111

Test Set of size $T$

## Batch Learning

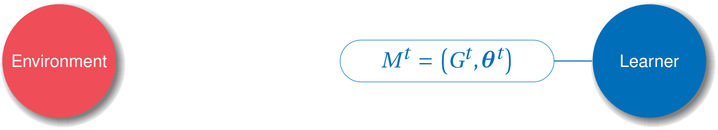
A *two-stage* process:

- A model $M \in \mathcal{M}$ is first extracted from a training set.
- The average loss of the model $M$ is evalued using a test set.


## Online Learning

A *sequential* process, or repeated game between the learner and its environment. During each trial $t = 1, \cdots, T$,

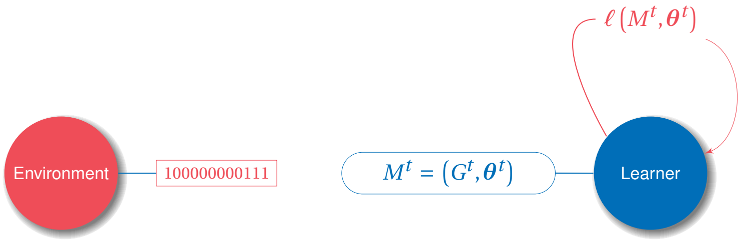
- the learner chooses a model $M^t \in \mathscr{M}$;
- the environment responds by an outcome $\boldsymbol{x}^t \in \mathscr{X}$, and the learner incurs the loss $\ell(M^t, x^t)$.




## Online Learning

A *sequential* process, or repeated game between the learner and its environment. During each trial $t = 1, \cdots, T$,

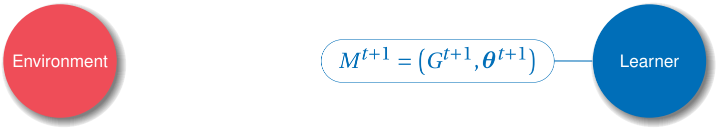
- the learner chooses a model $M^t \in \mathcal{M}$;
- the environment responds by an outcome $\boldsymbol{x}^t \in \mathcal{X}$, and the learner incurs the loss $\ell(M^t, x^t)$.




## Online Learning

A *sequential* process, or repeated game between the learner and its environment. During each trial $t = 1, \cdots, T$,

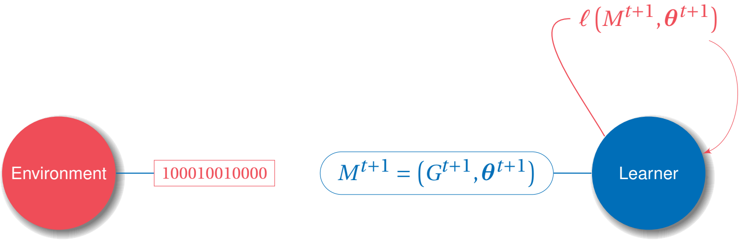
- the learner chooses a model $M^t \in \mathcal{M}$;
- the environment responds by an outcome $\boldsymbol{x}^t \in \mathcal{X}$, and the learner incurs the loss $\ell(M^t, x^t)$.




## Online Learning

A *sequential* process, or repeated game between the learner and its environment. During each trial $t = 1, \cdots, T$,

- the learner chooses a model $M^t \in \mathcal{M}$;
- the environment responds by an outcome $\boldsymbol{x}^t \in \mathcal{X}$, and the learner incurs the loss $\ell(M^t, x^t)$.




## Online Learning

A *sequential* process, or repeated game between the learner and its environment. During each trial $t = 1, \cdots, T$,

- the learner chooses a model $M^t \in \mathcal{M}$;
- the environment responds by an outcome $\boldsymbol{x}^t \in \mathcal{X}$, and the learner incurs the loss $\ell(M^t, x^t)$.

## Online vs. Batch

- In batch (PAC) learning, it is assumed that the data (training set + test set) is generated by a fixed (but unknown) probability distribution.
- In online learning, there is no statistical assumption about the data generated by the environment.

Online learning is particularly suited to:

* Adaptive environments, where the target distribution can change over time;
* Streaming applications, where all the data is not available in advance;
* Large-scale datasets, by processing only one outcome at a time.

## Online vs. Batch

- In batch (PAC) learning, it is assumed that the data (training set + test set) is generated by a fixed (but unknown) probability distribution.
- In online learning, there is no statistical assumption about the data generated by the environment.

Online learning is particularly suited to:

- Adaptive environments, where the target distribution can change over time;
- Streaming applications, where all the data is not available in advance;
- Large-scale datasets, by processing only one outcome at a time.

## Regret

Let $\mathscr{M}$ be a class of graphical models over an outcome space $\mathscr{X}$, and $A$ be a learning algorithm for $\mathscr{M}$.

The minimax regret of $A$ at horizon $T$, is the maximum, over every sequence of outcomes $\boldsymbol{x}^{1:T} = (x^1, \cdots, x^T)$, of the cumulative relative loss between $A$ and the best model in $\mathscr{M}$, i.e.

$$R(A,T) = \max_{\boldsymbol{x}^{1:T} \in \mathscr{X}^T} \left[ \sum_{t=1}^{T} \ell(M^t, \boldsymbol{x}^t) - \min_{M \in \mathscr{M}} \sum_{t=1}^{T} \ell(M, \boldsymbol{x}^t) \right]$$

## Learnability

A class $\mathscr{M}$ is (online) learnable if it admits an online learning algorithm $A$ such that:

1. the minimax regret of $A$ is sublinear in $T$, i.e.

$$\lim_{T \to \infty} R(A,T) = 0$$

2. the per-round computational complexity of $A$ is polynomial in the dimension of $\mathscr{M}$.

## Regret

Let $\mathscr{M}$ be a class of graphical models over an outcome space $\mathscr{X}$, and $A$ be a learning algorithm for $\mathscr{M}$.

The minimax regret of $A$ at horizon $T$, is the maximum, over every sequence of outcomes $\boldsymbol{x}^{1:T} = (x^1, \cdots, x^T)$, of the cumulative relative loss between $A$ and the best model in $\mathscr{M}$, i.e.

$$R(A, T) = \max_{\boldsymbol{x}^{1:T} \in \mathscr{X}^T} \left[ \sum_{t=1}^{T} \ell(M^t, \boldsymbol{x}^t) - \min_{M \in \mathscr{M}} \sum_{t=1}^{T} \ell(M, \boldsymbol{x}^t) \right]$$

## Learnability

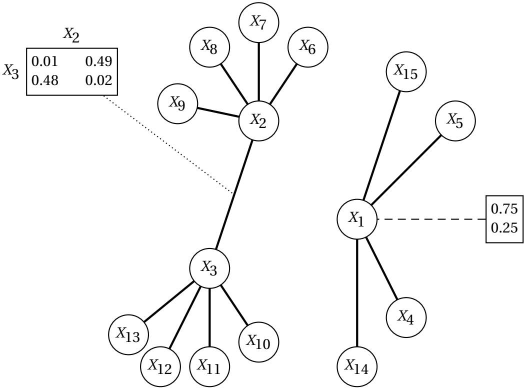
A class $\mathscr{M}$ is (online) learnable if it admits an online learning algorithm $A$ such that:

1. the minimax regret of $A$ is sublinear in $T$, i.e.

$$\lim_{T \to \infty} R(A, T) = 0$$

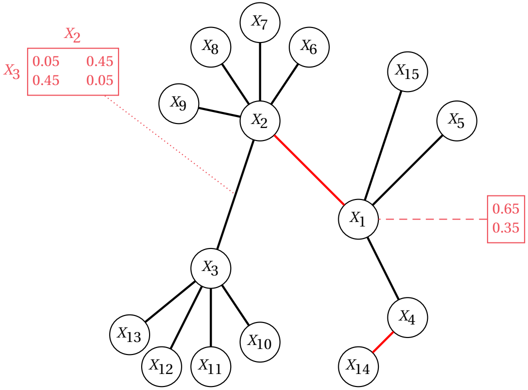
2. the per-round computational complexity of $A$ is polynomial in the dimension of $\mathscr{M}$.

# Outline




## Does there exist an efficient online learning algorithm for Markov forests?

How can we efficiently update at each iteration both the structure $F^t$ and the parameters $\boldsymbol{\theta}^t$ in order to minimize the cumulative loss $\sum_t \ell(F^t, \boldsymbol{\theta}^t, \boldsymbol{x}^t)$?




Does there exist an efficient online learning algorithm for Markov forests?

How can we efficiently update at each iteration both the structure $F^t$ and the parameters $\boldsymbol{\theta}^t$ in order to minimize the cumulative loss $\sum_t \ell(F^t, \boldsymbol{\theta}^t, \boldsymbol{x}^t)$?

### Two key properties

For the class $\mathscr{F}_{m,n} = \boldsymbol{F}_n \times \boldsymbol{\Theta}_{m,n}$ of Markov forests,

The probability distribution associated with a Markov forest $M = (F, \boldsymbol{\theta})$ can be factorized into a closed-form:

$$\mathbb{P}_M(\boldsymbol{x}) = \prod_{i=1}^{n} \theta_i(x_i) \prod_{(i,j)\in F} \frac{\theta_{ij}(x_i, x_j)}{\theta_i(x_i)\theta_j(x_j)}$$

The space $\boldsymbol{F}_n$ of forest structures is a matroid; minimizing a linear function over $\boldsymbol{F}_n$ can be done in quadratic time using the matroid greedy algorithm.

## Two key properties

For the class $\mathscr{F}_{m,n} = \boldsymbol{F}_n \times \boldsymbol{\Theta}_{m,n}$ of Markov forests,

The probability distribution associated with a Markov forest $M = (F, \boldsymbol{\theta})$ can be factorized into a closed-form:

$$\mathbb{P}_M(\boldsymbol{x}) = \prod_{i=1}^{n} \theta_i(x_i) \prod_{(i,j)\in F} \frac{\theta_{ij}(x_i, x_j)}{\theta_i(x_i)\theta_j(x_j)}$$

The space $\boldsymbol{F}_n$ of forest structures is a matroid; minimizing a linear function over $\boldsymbol{F}_n$ can be done in quadratic time using the matroid greedy algorithm.

## Log-Loss

Let $M = (\boldsymbol{f}, \boldsymbol{\theta})$ be a Markov forest, where $\boldsymbol{f}$ is the characteristic vector of the structure.

$$\begin{aligned}\ell(M, \boldsymbol{x}) &= -\ln \mathbb{P}_M(\boldsymbol{x}) \\ &= -\ln \left[ \prod_{i=1} \theta_i(x_i) \prod_{(i,j)} \left( \frac{\theta_{ij}(x_i, x_j)}{\theta_i(x_i)\theta_j(x_j)} \right)^{f_{ij}} \right]\end{aligned}$$

So, the log-loss is an affine function of the forest structure:

$$\ell(M, \boldsymbol{x}) = \psi(\boldsymbol{x}) + \langle \boldsymbol{f}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$$

where

$$\psi(\boldsymbol{x}) = \sum_{i \in [n]} \ln \frac{1}{\theta_i(x_i)} \quad \text{and} \quad \phi_{ij}(x_i, x_j) = \ln \left( \frac{\theta_i(x_i)\theta_j(x_j)}{\theta_{ij}(x_i, x_j)} \right)$$

## Log-Loss

Let $M = (\boldsymbol{f}, \boldsymbol{\theta})$ be a Markov forest, where $\boldsymbol{f}$ is the characteristic vector of the structure.

$$\begin{aligned}\ell(M, \boldsymbol{x}) &= -\ln \mathbb{P}_M(\boldsymbol{x}) \\ &= -\ln \left[ \prod_{i=1} \theta_i(x_i) \prod_{(i,j)} \left( \frac{\theta_{ij}(x_i, x_j)}{\theta_i(x_i)\theta_j(x_j)} \right)^{f_{ij}} \right]\end{aligned}$$

So, the log-loss is an affine function of the forest structure:

$$\ell(M, \boldsymbol{x}) = \psi(\boldsymbol{x}) + \langle \boldsymbol{f}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$$

where

$$\psi(\boldsymbol{x}) = \sum_{i \in [n]} \ln \frac{1}{\theta_i(x_i)} \quad \text{and} \quad \phi_{ij}(x_i, x_j) = \ln \left( \frac{\theta_i(x_i)\theta_j(x_j)}{\theta_{ij}(x_i, x_j)} \right)$$

## Regret Decomposition

Based on the linearity of the log-loss, the regret is decomposable into two parts:

$$R\left(M^{1:T}, \boldsymbol{x}^{1:T}\right) = R\left(\boldsymbol{f}^{1:T}, \boldsymbol{x}^{1:T}\right) + R\left(\boldsymbol{\theta}^{1:T}, \boldsymbol{x}^{1:T}\right)$$

where

$$R\left(\boldsymbol{f}^{1:T}, x^{1:T}\right) = \sum_{t=1}^{T} \ell(\boldsymbol{f}^{t}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) - \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) \qquad \text{(Structural Regret)}$$

$$R\left(\boldsymbol{\theta}^{1:T}, x^{1:T}\right) = \sum_{t=1}^{T} \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) - \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{*}, \boldsymbol{x}^{t}) \qquad \text{(Parametric Regret)}$$

## Regret Decomposition

Based on the linearity of the log-loss, the regret is decomposable into two parts:

$$R\left(M^{1:T}, \boldsymbol{x}^{1:T}\right) = R\left(\boldsymbol{f}^{1:T}, \boldsymbol{x}^{1:T}\right) + R\left(\boldsymbol{\theta}^{1:T}, \boldsymbol{x}^{1:T}\right)$$

where

$$R\left(\boldsymbol{f}^{1:T}, x^{1:T}\right) = \sum_{t=1}^{T} \ell(\boldsymbol{f}^{t}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) - \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) \qquad \text{(Structural Regret)}$$

$$R\left(\boldsymbol{\theta}^{1:T}, x^{1:T}\right) = \sum_{t=1}^{T} \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{t}, \boldsymbol{x}^{t}) - \ell(\boldsymbol{f}^{*}, \boldsymbol{\theta}^{*}, \boldsymbol{x}^{t}) \qquad \text{(Parametric Regret)}$$

## Parametric Regret

Based on the closed-form expression of Markov forests, the parametric regret is decomposable into local regrets:

$$R\left(\boldsymbol{\theta}^{1:T}, x^{1:T}\right) = \sum_{i=1}^{n} \ln \frac{\theta_i^*(x_i^{1:T})}{\theta_i^{1:T}(x_i^{1:T})} \qquad \text{(Univariate estimators)}$$

$$+ \sum_{(i,j)\in F} \ln \frac{\theta_{ij}^*(x_{ij}^{1:T})}{\theta_{ij}^{1:T}(x_{ij}^{1:T})} \qquad \text{(Bivariate estimators)}$$

$$+ \sum_{(i,j)\in F} \ln \frac{\theta_i^{1:T}(x_i^{1:T})}{\theta_i^*(x_i^{1:T})} \frac{\theta_j^{1:T}(x_j^{1:T})}{\theta_j^*(x_j^{1:T})} \qquad \text{(Bivariate compensation)}$$

where

$$\theta_i^*(x_i^{1:T}) = \prod_{t=1}^{T} \theta_i^*(x_i^t), \qquad \theta_i^{1:T}(x_i^{1:T}) = \prod_{t=1}^{T} \theta_i^t(x_i^t)$$

$$\theta_{ij}^*(x_{ij}^{1:T}) = \prod_{t=1}^{T} \theta_{ij}^*(x_i^t, x_j^t), \qquad \theta_{ij}^{1:T}(x_{ij}^{1:T}) = \prod_{t=1}^{T} \theta_{ij}^t(x_i^t, x_j^t)$$

## Parametric Regret

Based on the closed-form expression of Markov forests, the parametric regret is decomposable into local regrets:

$$
\begin{aligned}
R\left(\boldsymbol{\theta}^{1:T}, x^{1:T}\right) = &\sum_{i=1}^{n} \ln \frac{\theta_i^*(x_i^{1:T})}{\theta_i^{1:T}(x_i^{1:T})} && \text{(Univariate estimators)}\\
&+ \sum_{(i,j)\in F} \ln \frac{\theta_{ij}^*(x_{ij}^{1:T})}{\theta_{ij}^{1:T}(x_{ij}^{1:T})} && \text{(Bivariate estimators)}\\
&+ \sum_{(i,j)\in F} \ln \frac{\theta_i^{1:T}(x_i^{1:T})}{\theta_i^*(x_i^{1:T})} \frac{\theta_j^{1:T}(x_j^{1:T})}{\theta_j^*(x_j^{1:T})} && \text{(Bivariate compensation)}
\end{aligned}
$$

where

$$
\theta_i^*(x_i^{1:T}) = \prod_{t=1}^{T} \theta_i^*(x_i^t), \qquad \theta_i^{1:T}(x_i^{1:T}) = \prod_{t=1}^{T} \theta_i^t(x_i^t)
$$

$$
\theta_{ij}^*(x_{ij}^{1:T}) = \prod_{t=1}^{T} \theta_{ij}^*(x_i^t, x_j^t), \qquad \theta_{ij}^{1:T}(x_{ij}^{1:T}) = \prod_{t=1}^{T} \theta_{ij}^t(x_i^t, x_j^t)
$$

## Local regrets

Expressions of the form

$$\frac{\theta^*(x^{1:T})}{\theta^{1:T}(x^{1:T})}$$

Use Dirichlet Mixtures

have been extensively studied in literature of universal coding (Grünwald, 2007).

## Dirichlet Mixtures

Using symmetric Dirichlet mixtures for the parametric estimators,

$$\theta^{1:T}(x^{1:T}) = \int \prod_{t=1}^{T} \mathbb{P}_{\boldsymbol{\lambda}}(x^t) p_\mu(\boldsymbol{\lambda})\, d\boldsymbol{\lambda} = \frac{\Gamma(m\mu)}{\Gamma(\mu)^m} \frac{\prod_{v=1}^{m} \Gamma(t_v+\mu)}{\Gamma(t+m\mu)}$$

If $\mu = \frac{1}{2}$, we get the Jeffreys mixture.

## Local regrets

Expressions of the form

$$\frac{\theta^*(x^{1:T})}{\theta^{1:T}(x^{1:T})}$$

Use Dirichlet Mixtures

have been extensively studied in literature of universal coding (Grünwald, 2007).

## Dirichlet Mixtures

Using symmetric Dirichlet mixtures for the parametric estimators,

$$\theta^{1:T}(x^{1:T}) = \int \prod_{t=1}^{T} \mathbb{P}_{\boldsymbol{\lambda}}(x^t) p_\mu(\boldsymbol{\lambda})\, d\boldsymbol{\lambda} = \frac{\Gamma(m\mu)}{\Gamma(\mu)^m} \frac{\prod_{v=1}^{m} \Gamma(t_v + \mu)}{\Gamma(t + m\mu)}$$

If $\mu = \frac{1}{2}$, we get the Jeffreys mixture.

## Local regrets

Expressions of the form

$$\frac{\theta^*(x^{1:T})}{\theta^{1:T}(x^{1:T})}$$

Use Dirichlet Mixtures

have been extensively studied in literature of universal coding (Grünwald, 2007).

## Dirichlet Mixtures

Using symmetric Dirichlet mixtures for the parametric estimators,

$$\theta^{1:T}(x^{1:T}) = \int \prod_{t=1}^{T} \mathbb{P}_{\boldsymbol{\lambda}}(x^t) p_\mu(\boldsymbol{\lambda}) d\boldsymbol{\lambda} = \frac{\Gamma(m\mu)}{\Gamma(\mu)^m} \frac{\prod_{v=1}^{m} \Gamma(t_v + \mu)}{\Gamma(t + m\mu)}$$

If $\mu = \frac{1}{2}$, we get the Jeffreys mixture.

### Jeffreys Strategy (An extension of Xie and Barron (2000) to forest parameters)

For each trial $t$,

1. Set $\theta_i^{t+1}(u) = \dfrac{t_u + \frac{1}{2}}{t + \frac{m}{2}}$ for all $i \in [n], u \in [m]$
2. Set $\theta_{ij}^{t+1}(u,v) = \dfrac{t_{uv} + \frac{1}{2}}{t + \frac{m^2}{2}}$ for all $(i,j) \in \binom{[n]}{2}, u, v \in [m]$

### Performance

- Minimax regret: $\dfrac{n(m-1) + (n-1)(m-1)^2}{2} \ln \frac{T}{2\pi} + C_{m,n} + o(m^2 n)$
- Per-round time complexity: $O(m^2 n^2)$

## Jeffreys Strategy (An extension of Xie and Barron (2000) to forest parameters)

For each trial $t$,

1. Set $\theta_i^{t+1}(u) = \dfrac{t_u + \frac{1}{2}}{t + \frac{m}{2}}$ for all $i \in [n], u \in [m]$
2. Set $\theta_{ij}^{t+1}(u, v) = \dfrac{t_{uv} + \frac{1}{2}}{t + \frac{m^2}{2}}$ for all $(i, j) \in \binom{[n]}{2}, u, v \in [m]$

## Performance

- Minimax regret: $\dfrac{n(m-1) + (n-1)(m-1)^2}{2} \ln \frac{T}{2\pi} + C_{m,n} + o(m^2 n)$
- Per-round time complexity: $O(m^2 n^2)$

## Structural Regret

Based on the linear form of the log-loss, the structural learning problem can be cast as a sequential combinatorial optimization problem (Audibert et al., 2011).

Minimize

$$\sum_{t=1}^{T}\left[\sum_{s=1}^{t}\langle \boldsymbol{f}^s, \phi(\boldsymbol{x}^s)\rangle\right]$$

subject to

$$\boldsymbol{f}_1, \cdots, \boldsymbol{f}_T \in \boldsymbol{F}_n$$

## Structural Regret

Based on the linear form of the log-loss, the structural learning problem can be cast as a sequential combinatorial optimization problem (Audibert et al., 2011).

Minimize

$$\sum_{t=1}^{T}\left[\sum_{s=1}^{t}\langle \boldsymbol{f}^{s}, \phi(\boldsymbol{x}^{s})\rangle\right]$$

Use the greedy algorithm

subject to

$$\boldsymbol{f}_1, \cdots, \boldsymbol{f}_T \in \boldsymbol{F}_n$$

### Follow the Perturbed Leader (Kalai and Vempala, 2005)

For each trial $t$,

1. Draw $\boldsymbol{r}_t$ in $\left[0, \frac{1}{\alpha_t}\right]$ uniformly at random
2. Set $\boldsymbol{f}^{t+1} = \mathrm{argmin}_{\boldsymbol{f} \in \boldsymbol{F}_n} \langle \boldsymbol{f}, \boldsymbol{L}^t + r^t \rangle$

where $L^t = \sum_{s=1}^{t} \phi(\boldsymbol{x}^s)$

### Performance

- Minimax regret: $n^2 \ln(T/2 + m^2/4)\sqrt{2T}$
- Per-round time complexity: $O(n^2 \log n)$

## Follow the Perturbed Leader (Kalai and Vempala, 2005)

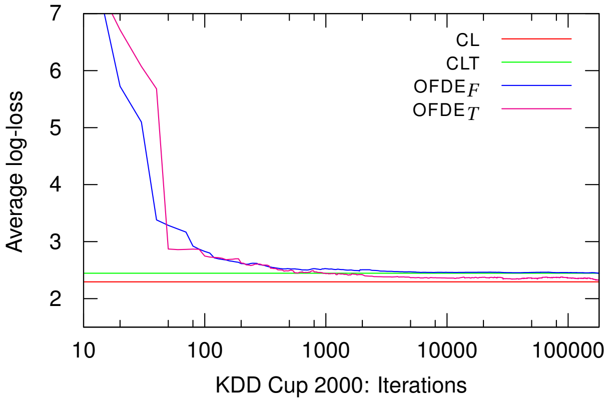
For each trial $t$,

1. Draw $\boldsymbol{r}_t$ in $\left[0, \frac{1}{\alpha_t}\right]$ uniformly at random
2. Set $\boldsymbol{f}^{t+1} = \text{argmin}_{\boldsymbol{f} \in \boldsymbol{F}_n} \langle \boldsymbol{f}, \boldsymbol{L}^t + r^t \rangle$

where $L^t = \sum_{s=1}^{t} \phi(\boldsymbol{x}^s)$

## Performance

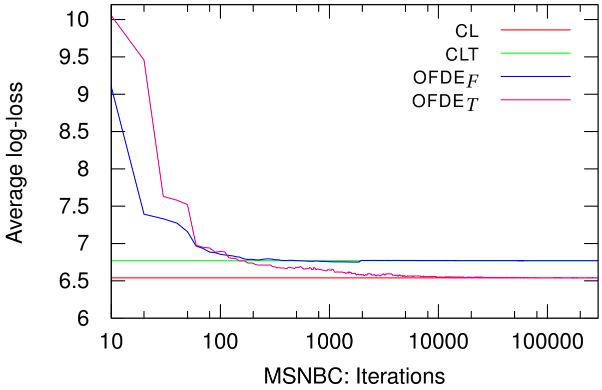
- Minimax regret: $n^2 \ln(T/2 + m^2/4)\sqrt{2T}$
- Per-round time complexity: $O(n^2 \log n)$




## Experiments

The average log-loss is measured on the test set at each iteration.

The online learner (FPL + Jeffreys) rapidly converges to the batch learner (Chow-Liu for Markov trees, and Thresholded Chow-Liu for Markov forests).




## Experiments

The average log-loss is measured on the test set at each iteration.

The online learner (FPL + Jeffreys) rapidly converges to the batch learner (Chow-Liu for Markov trees, and Thresholded Chow-Liu for Markov forests).