SEMINARIO DEL 15/05

Ruggieri Andrea

Stranieri Francesco

MAD Lab

INTRODUZIONE

EM HARD vs EM SOFT

EM HARD VS EM SOFT

Ultimo meeting

	EM SOFT	EM HARD
Inizializzazione parametri	Random, uniforme o qualsiasi altro modo	Random, uniforme o qualsiasi altro modo
Expectation	Si prendono in considerazione tutte le possibili combinazioni dei dati	Si calcolano tutte le possibile combinazioni di assegnamento di valori alle variabili ma si sceglie il singolo assegnamento che massimizza la joint distribution *(max)*
Expected sufficient statistics	$\bar{M}_\theta[u] = \sum_{m=1}^{M} \sum_{h[m] \epsilon Val(H[m])} Q(h[m]) \mathrm{I}\{\xi[m] < Y > = y\}$	$\bar{M}_{\theta^t}[x^1] = \sum_{m}^{M} \boldsymbol{I}\{\xi[m] < X > = x^1\}$
Maximisation step	Sulla base delle Expected Sufficient Statistics si aggiornano i parametri	Sulla base delle Expected Sufficient Statistics si aggiornano i parametri

EM HARD VS EM SOFT

Ultimo meeting

- Entrambi i metodi eseguono due step: Completare i dati utilizzando i parametri θ^t e usare questi dati per computare i nuovi parametri θ^{t+1}

- Tuttavia, diversamente da EM SOFT, EM HARD seleziona, per ogni istanza o[m] il singolo assegnamento h[m] che massimizza $P(h|o[m], \theta^t)$

- EM HARD può essere visto come l'ottimizzazione di una diversa funzione obiettivo che coinvolge sia l'apprendimento di parametri sia il compito di apprendere il corretto assegnamento delle variabili mancanti. **L'obiettivo è quello di massimizzare la likelihood dei dati completi dati i parametri**

$$\max_{\theta,H} l(\theta: H, D)$$

- EM SOFT al contrario, tenta di massimizzare $l(\theta: D)$, **considerando tutti i possibili assegnamenti dei dati mancanti**

TABLE OF CONTENTS

In questa presentazione si tratteranno i seguenti argomenti:

EM HARD

Implementazione

EM HARD – IMPLEMENTAZIONE

Lo sviluppo di EM HARD si è basato sullo script EM SOFT

Opportune modifiche sono state effettuate al metodo `maximisation_step` in fase di computazione delle **expected sufficient statistics**.

EM HARD non aggiunge nessuna struttura dati e nessun nuovo metodo. Rispetto a EM SOFT risulta essere **memory-friendly** e computazionalmente meno complesso

La computazione delle **expected sufficient statistics** si basa sul concetto di **combinazione più probabile**

EM HARD – IMPLEMENTAZIONE

$Q^1(< B^T, A^T >) = \alpha(0{,}1 * 0{,}8 * 0{,}6 * 0{,}1 * 0{,}2) = \alpha(9{,}6 \cdot 10^{-4}) = 2{,}17 \cdot 10^{-3}$

$Q^1(< B^T, A^F >) = \alpha(0{,}1 * 0{,}8 * 0{,}4 * 0{,}8 * 0{,}9) = \alpha(0{,}02304) = 0{,}05217$

$Q^1(< B^F, A^T >) = \alpha(0{,}9 * 0{,}8 * 0{,}2 * 0{,}1 * 0{,}2) = \alpha(2{,}88 \cdot 10^{-3}) = 6{,}52 * 10^{-3}$

$Q^1(< B^F, A^F >) = \alpha(0{,}9 * 0{,}8 * 0{,}8 * 0{,}8 * 0{,}9) = \alpha(0{,}4147) = 0{,}939$

$Q^2(< E^T, A^T >) = \alpha(0{,}9 * 0{,}2 * 0{,}3 * 0{,}9 * 0{,}2) = \alpha(9{,}72 \cdot 10^{-3}) = 0{,}006$

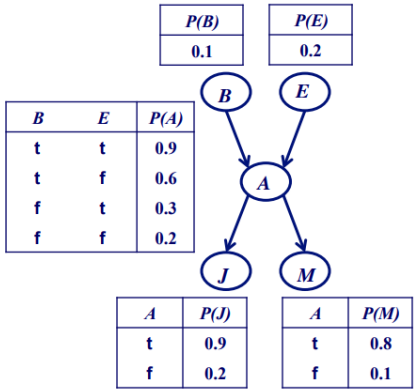
$Q^2(< E^T, A^F >) = \alpha(0{,}9 * 0{,}2 * 0{,}7 * 0{,}2 * 0{,}9) = \alpha(0{,}02268) = 0{,}14$

$Q^2(< E^F, A^T >) = \alpha(0{,}9 * 0{,}8 * 0{,}2 * 0{,}9 * 0{,}2) = \alpha(0{,}02592) = 0{,}16$

$Q^2(< E^F, A^F >) = \alpha(0{,}9 * 0{,}8 * 0{,}8 * 0{,}2 * 0{,}9) = \alpha(0{,}10368) = 0{,}64$

$Q^3(< J^T >) = \alpha(0{,}9 * 0{,}2 * 0{,}3 * 0{,}9 * 0{,}2) = \alpha(9{,}72 \cdot 10^{-3}) = 0{,}9$

$Q^3(< J^F >) = \alpha(0{,}9 * 0{,}2 * 0{,}3 * 0{,}1 * 0{,}2) = \alpha(1{,}08 \cdot 10^{-3}) = 0{,}1$



	D1	D2	D3
B	?	F	F
E	F	?	T
A	?	?	T
J	F	T	?
M	F	F	F

EM HARD – IMPLEMENTAZIONE

Input per calcolo delle nuove CPT

	D1	D2	D3
B	F	F	F
E	F	F	T
A	F	F	T
J	F	T	T
M	F	F	F

Le CPT vengono ora computate considerando il dataset come completo

EM HARD – IMPLEMENTAZIONE

***Possibile task per il futuro**: ottimizzare ancora di più il codice per cercare di ridurre i tempi computazionali e lo spazio richiesto in memoria*

LIBRERIA

Struttura finale

ARCHITETTURA

L’esecuzione dell’algoritmo EM HARD parte chiamando il metodo `em_hard`. L’esecuzione dell’algoritmo EM SOFT parte chiamando il metodo `em_soft`. Gli script richiedono i seguenti parametri in input:

- **initial_data** - *OBBLIGATORIO* - Si tratta di un dataframe dove è presente almeno un missing value. Si consiglia di passare in input all'interno del metodo dataframe che hanno come intestazioni delle righe il nome delle variabili; mentre, come intestazione delle colonne, i dati. Si consiglia inoltre di passare in input all'algoritmo dataframe numerici. In caso di valori nominali, si consiglia di convertire i valori in numerico
- **structure** - *OBBLIGATORIO* - La struttura della rete bayesiana. Si prega di definire la struttura della rete attraverso `model2network` di bnlearn
- **cpt** - *OBBLIGATORIO* - Si tratta dell'inizializzazione iniziale date alle CPT della rete. L'inizializzazione può essere uniforme, randomica o di qualsiasi altra natura
- **ALPHA** - *FACOLTATIVO* - iperparametro che sottende lo stopping criteria. Se la differenza in valore assoluto tra le cpt calcolate all'iterazione corrente con le cpt calcolate all'iterazione precedente è minore di questo valore, l'algoritmo si arresta e termina la sua computazione. `ALPHA` deve essere definito positivo. Default ALPHA = 0.05
- **NUMBER_ITERACTION** - *FACOLTATIVO* - iperparametro che indica il massimo numero di iterazioni che devono essere effettuate se l'algoritmo non si è arrestato attraverso la politica di stopping criteria. Default `NUMBER_ITERACTION` = 10

ARCHITETTURA

I metodi `em_soft` *e* `em_hard` *hanno la seguente struttura*

Controllo delle variabili in input

Preprocessing e preparazione all'esecuzione dell'algoritmo

Esecuzione dell'algoritmo expectation maximisation

ARCHITETTURA

I metodi `em_soft` *e* `em_hard` *hanno la seguente struttura*

Controllo delle variabili in input

- Controllato che tutti i parametri obbligatori sono stati passati correttamente
- Si controlla che il dataset sia stato passato correttamente. Se il dataset non è nella forma desiderata, lo script prova a trasformarlo
  - Nelle righe devono comparire i nomi delle variabili mentre nelle colonne i dati
  - Dataset deve essere numerico
- Viene controllato se le CPT sono state passate correttamente implementando diversi controlli

Preprocessing e preparazione all'esecuzione dell'algoritmo

Esecuzione dell'algoritmo expectation maximisation

ARCHITETTURA

I metodi `em_soft` *e* `em_hard` *hanno la seguente struttura*

Controllo delle variabili in input

Preprocessing e preparazione all'esecuzione dell'algoritmo

- Viene calcolato il numero di missing data presenti nel dataset andando a memorizzare tutti gli indici dei dati con valori mancanti
- Si assegna una distribuzione uniforma di probabilità iniziale per tutte le variabili missing
- Si crea la tabella `table_posterior_prob` presentata nelle presentazioni precedenti

Esecuzione dell'algoritmo expectation maximisation

ARCHITETTURA

I metodi `em_soft` e `em_hard` hanno la seguente struttura

Controllo delle variabili in input

Preprocessing e preparazione all'esecuzione dell'algoritmo

Esecuzione dell'algoritmo expectation maximisation

Si itera da 1 a `NUMBER_ITERACTION`

- Expectation step
- Maximisation step
- Stopping criteria

Tutti questi step sono stati presentati in maniera approfondita nelle presentazioni precedenti

ARCHITETTURA

I metodi `em_soft` *e* `em_hard` *hanno la seguente struttura*

Controllo delle variabili in input

Preprocessing e preparazione all'esecuzione dell'algoritmo

Esecuzione dell'algoritmo Expectation Maximisation

Si itera da 1 a `NUMBER_ITERACTION`

- Expectation step
- Maximisation step
- Stopping criteria

Tutti questi step sono stati presentati in maniera approfondita nelle presentazioni precedenti

Passo più costoso dal punto di vista computazionale

CONFRONTO DEI RISULTATI

Al termine dell'esecuzione finale degli script viene restituito in output l'intero dataset con i valori rimpiazzati.

Sono state messe a disposizione due funzioni per il confronto dei risultati. Queste funzioni sono presenti nello script EM_COMPARISON

- *compare_em* – Prende in input i due dataset ottenuti dalle esecuzioni di EM HARD e EM SOFT e il dataset iniziale contenente i missing value. Restituisce in output la percentuale dei valori rimpiazzati allo stesso modo da EM HARD e da EM SOFT

- *compare_em_with_ground_truth* – Prende in input un dataset ottenuto (EM HARD o EM SOFT), il dataset iniziale contenente i missing values e un dataset completo che fungerà da ground truth. Restitituisce in output la percentuale di valori rimpiazzati correttamente dal dataset originato dall'esecuzione dello script EM

TEST ESEGUITI

Introduzione ai test

ASIA

INTRODUZIONE AI TEST

Sono stati condotti diversi test ai fini di valutare la correttezza degli script e risolvere eventuali bugs. La maggior parte di questi test sono stati condotti su dataset molto piccoli, che non presenteremo in questa sessione. In questa parte esamineremo nel dettaglio i test condotti su due dataset: ASIA e ALARM

INTRODUZIONE AI TEST

Concetti di base

Notazione: Chiamiamo **PROP (Proportion of missing data)** l'iperparametro che indica la percentuale di missing data presenti nel dataset

Esempio: se un dataset consiste di 5.000 dati, allora PROP = 0,3 implica che 1.500 dati conteranno almeno un missing value

INTRODUZIONE AI TEST

Concetto di EM SOFT FORCED

Oltre alle versioni: EM HARD e EM SOFT è stata presa in considerazione una terza implementazione dell'algoritmo EM che prende il nome di **EM SOFT FORCED**. Si tratta dello stesso algoritmo implementato in EM SOFT ma **viene forzato il numero di iterazioni ad essere uguali ad EM HARD.**

Se EM HARD termina la sua computazione al quarto step, EM SOFT FORCED deve terminare al quarto step mentre EM SOFT continua la sua esecuzione per più iterazioni

INTRODUZIONE AI TEST

Aspetti analizzati:

C'è differenza tra EM SOFT e EM HARD in termini di accuratezza dei risultati?

Che **impatto** ha l'algoritmo EM nei **tempi computazionali**?

Che risultati si ottengono forzando EM SOFT a **terminare nello stesso numero di iterazioni** di EM HARD?

Come variano le performance dell'algoritmo EM al variare del **tipo di dati mancanti?**

Dati missing generati utilizzando il metodo `ampute` messo a disposizione dalla libreria **mice** di R

ASIA

Un esempio classico di DBN è il network **ASIA** [Lauritzen & Spiegelhalter, 1998] che include una raccolta di variabili binarie.

La rete descrive un semplice problema per la **diagnosi della tubercolosi e del cancro ai polmoni**:

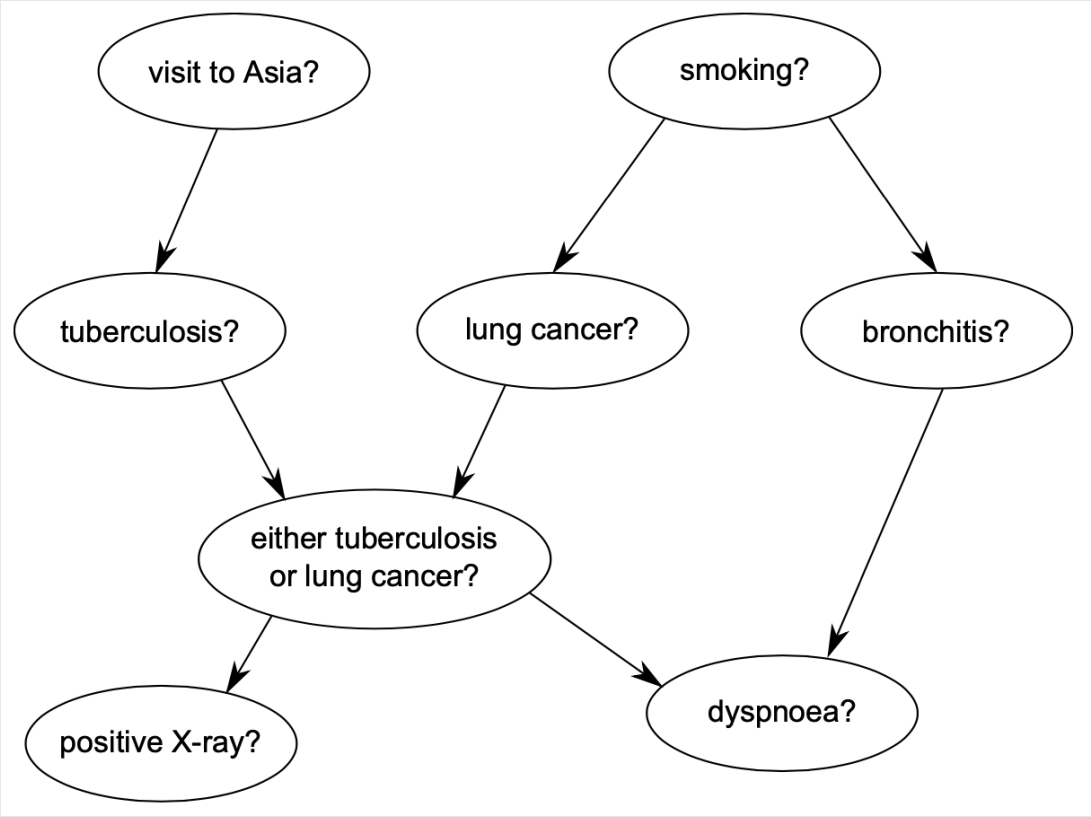
"La mancanza di respiro (dispnea) può essere dovuta a tubercolosi, cancro ai polmoni o bronchite, o a nessuno di essi, o a più di uno di essi.

Una recente visita in Asia aumenta le probabilità di contrarre la tubercolosi, mentre è noto che il fumo è un fattore di rischio sia per il cancro ai polmoni che per la bronchite.

I risultati di una singola radiografia del torace non discriminano tra cancro ai polmoni e tubercolosi, così come la presenza o l'assenza di dispnea".

ASIA

La rete descrive un semplice problema per la diagnosi della tubercolosi e del cancro ai polmoni.

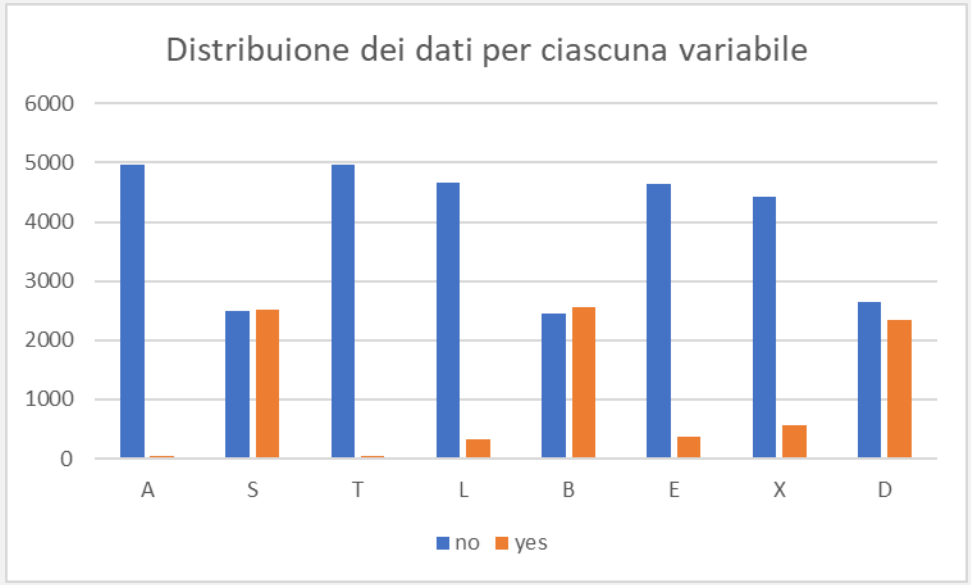


Numero di nodi: 8
Numero di archi: 8
Numero di parametri: 18

ASIA

Il dataset è composto da 5.000 dati e 40.000 celle

A	S	T	L	B	E	X	D
no :4958	no :2485	no :4956	no :4670	no :2451	no :4630	no :4431	no :2650
yes: 42	yes:2515	yes: 44	yes: 330	yes:2549	yes: 370	yes: 569	yes:2350



Ci sono tante variabili **sbilanciate** e per alcune di queste variabili, la classe di minoranza risulta essere estremamente rara

ASIA

Generiamo dati **MNAR** attraverso il metodo `ampute` e consideriamo solo 2720 dati

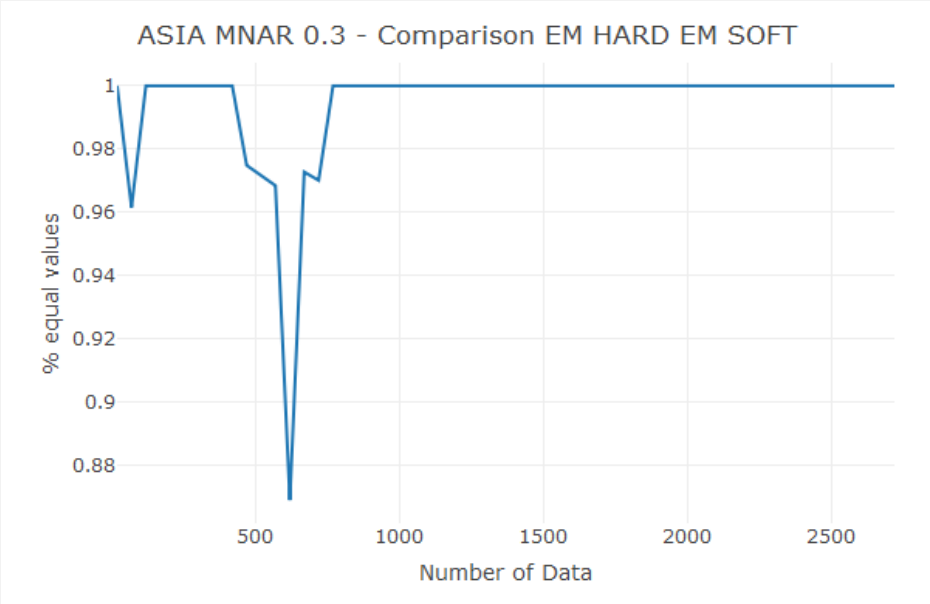
Consideriamo 2 contesti: PROP = 0.3 e PROP = 0.6

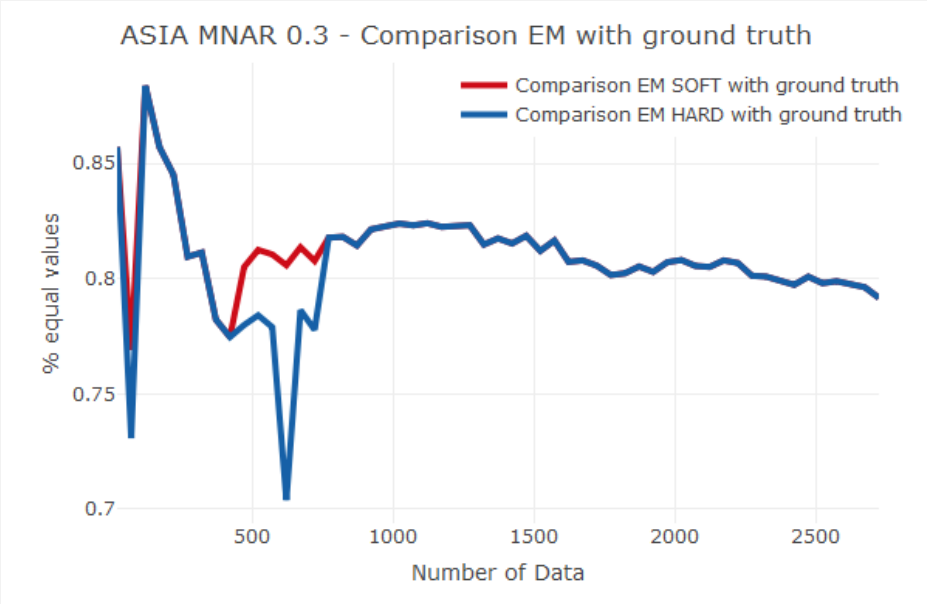
ASIA

Missing Not at Random (MNAR)

PROP 0.3

FORCED



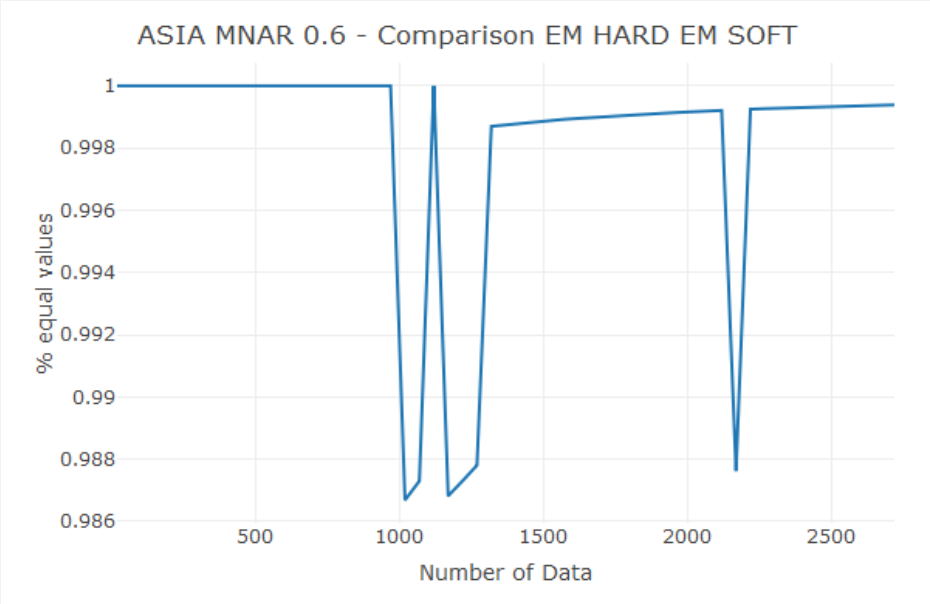


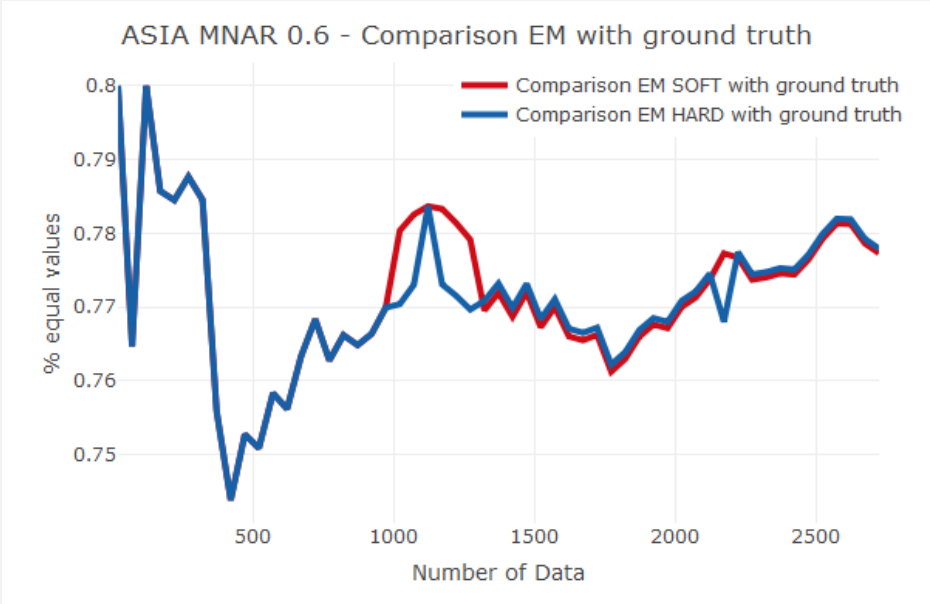
ASIA

Missing Not at Random (MNAR)

PROP 0.6

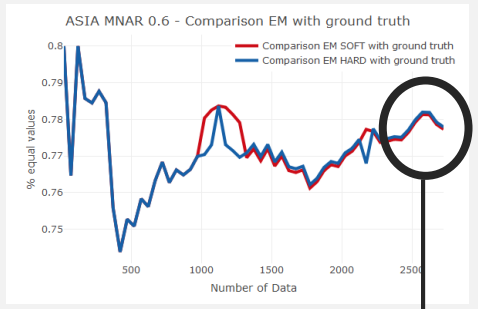
FORCED

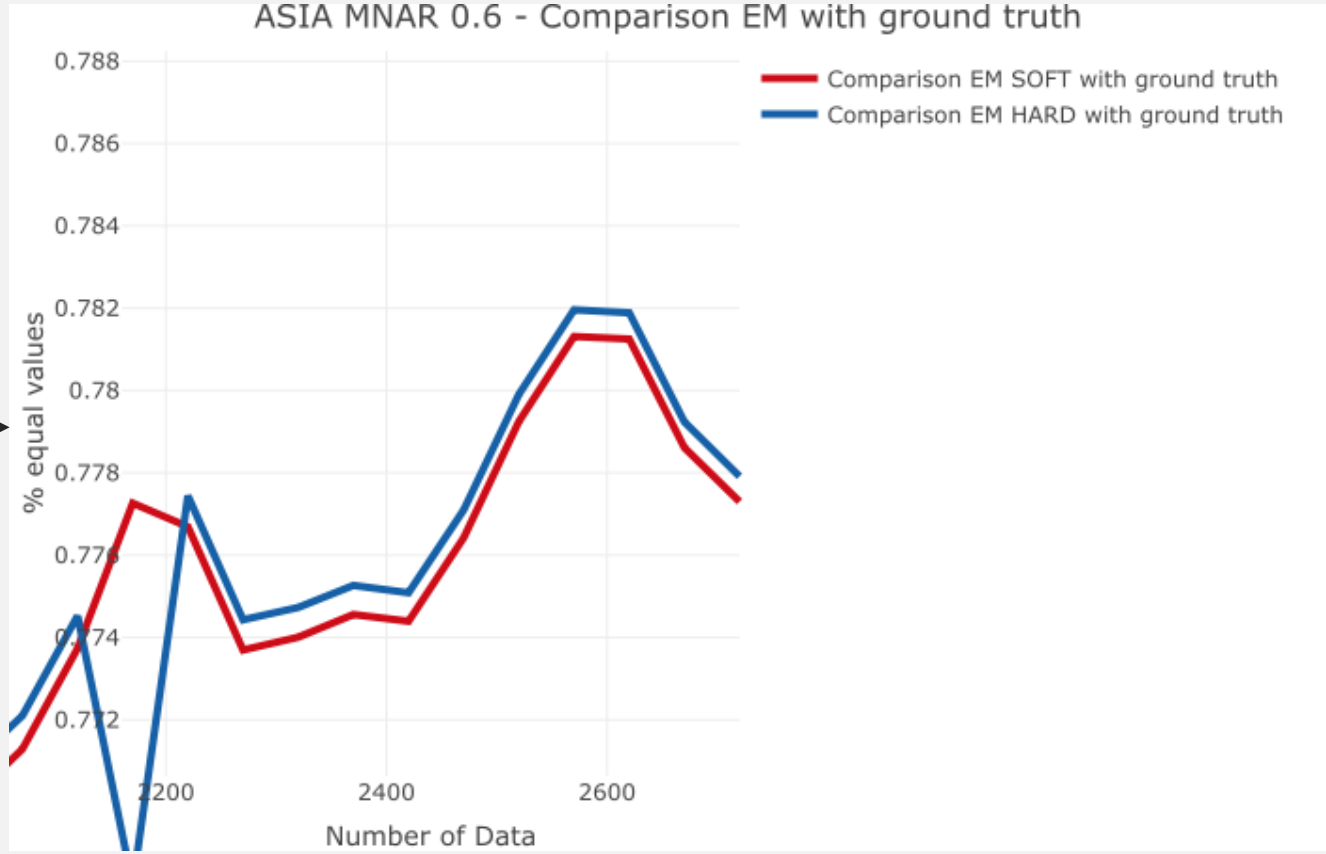




ASIA

Missing Not at Random (MNAR)

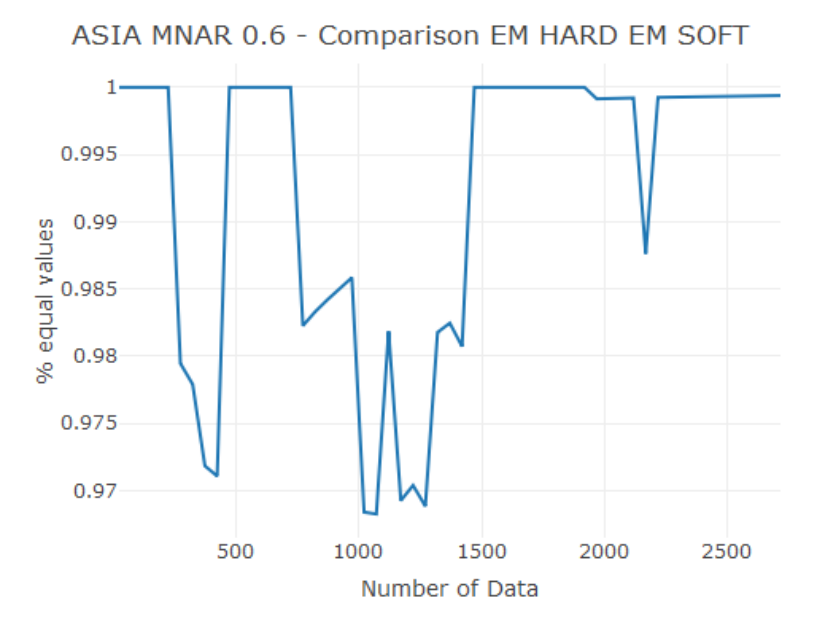


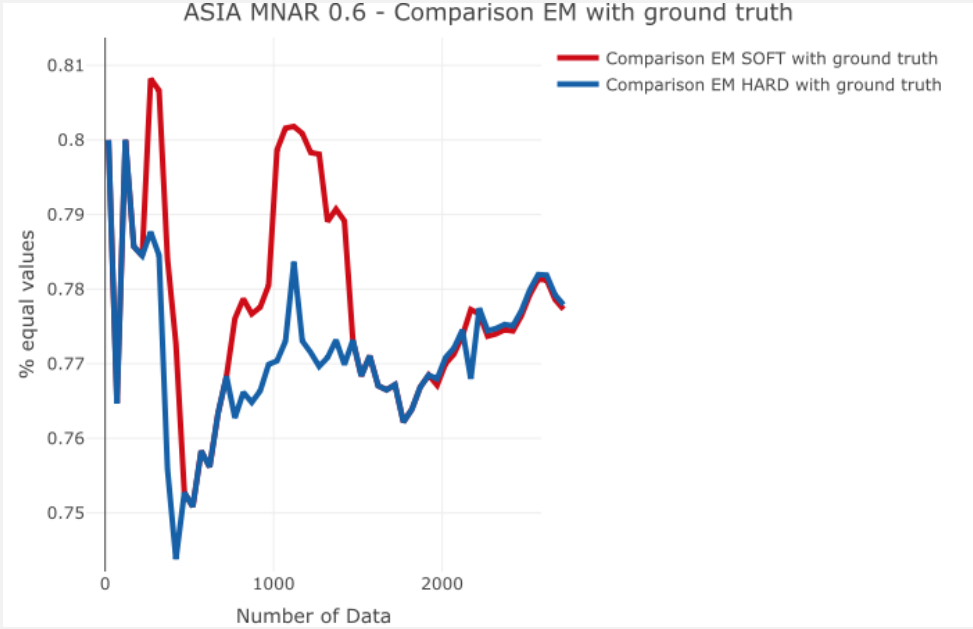


ASIA

Missing Not at Random (MNAR)

PROP 0.6

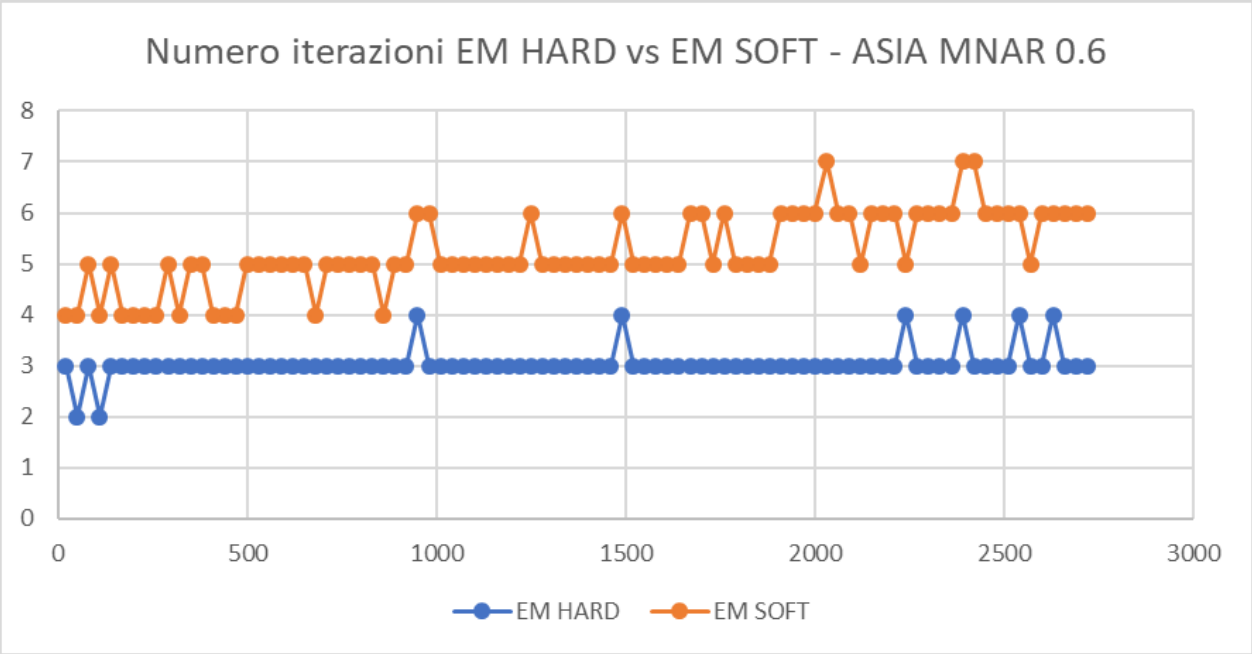




ASIA

Missing Not at Random (MNAR)

PROP 0.6



ASIA

Fino a questo punto ci siamo limitati a simulazioni che andavano da 20 dati a 2720 dati. Cosa succede se eseguiamo EM HARD e EM SOFT su ***tutto il dataset****?*

ASIA

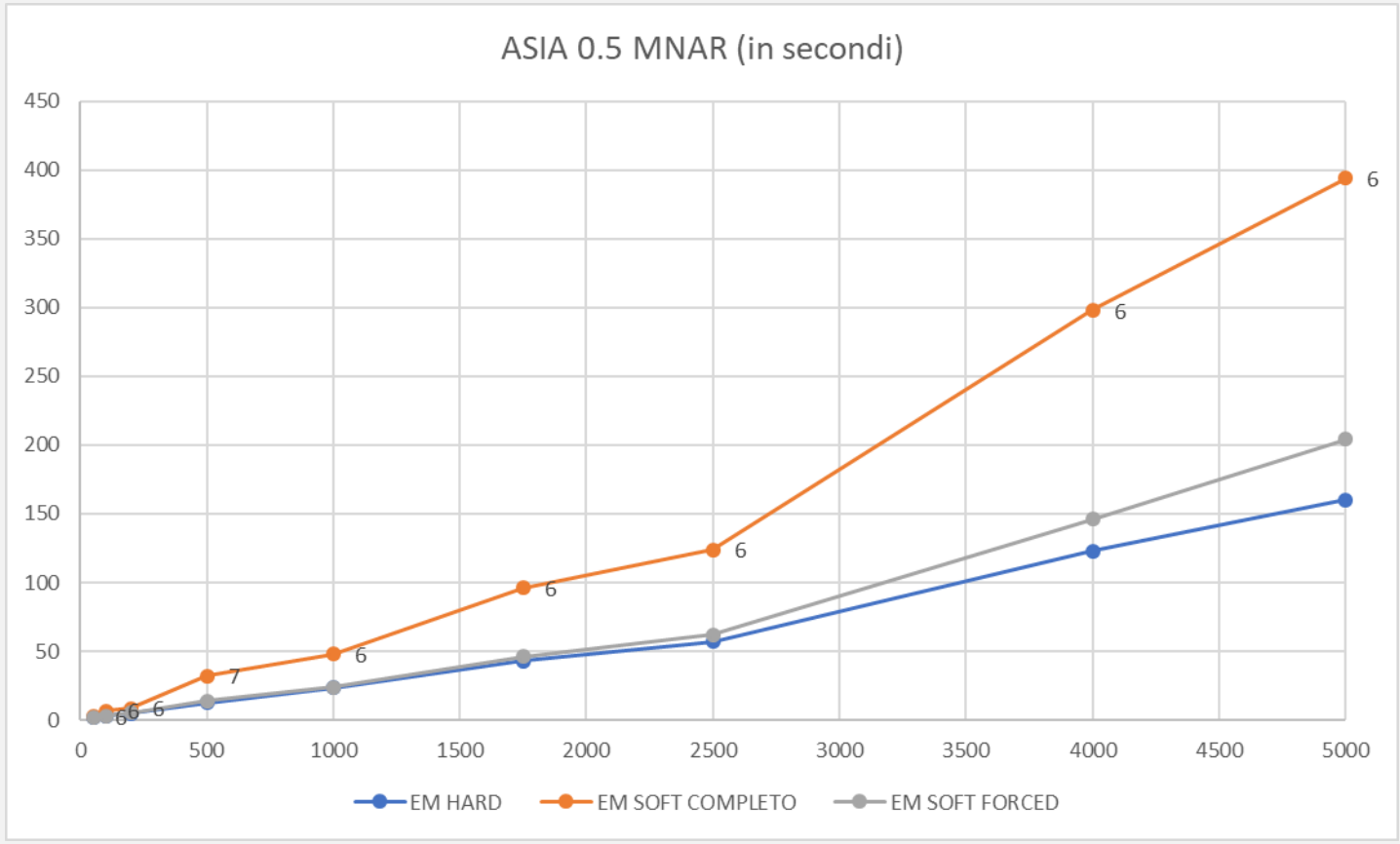
Fino a questo punto ci siamo limitati a simulazioni che andavano da 20 dati a 2720 dati. Cosa succede se eseguiamo EM HARD e EM SOFT su tutto il dataset?

TIPO DI DATI	ALPHA	PROP	Numero missing	Iterazioni EM HARD	Iterazioni EM SOFT	Tempo EM HARD	Tempo EM SOFT	EM HARD /EM SOFT	EM HARD /Ground TRUTH	EM SOFT/Ground TRUTH
MNAR	0,05	0,2	941	3	5	89 sec	185 sec	1	0,7768332	0,7768332
MNAR	0,05	0,5	2450	3	6	160 sec	394 sec	1	0,7587755	0,758755
MNAR	0,05	0,8	4031	3	7	262 sec	680 sec	0,995	0,8082362	0,8112131
MNAR	0,05	0,5	2450	3	3(forced)	160 sec	204 sec	0,9987755	**0,7587755**	**0,757551**

Tabella computata su tutti i 5000 dati presenti nel dataset

ASIA

Vediamo ora come si comportano EM HARD ed EM SOFT in termini di tempi computazionali



ASIA

Per finire, fissiamo il numero di dati pari a 1000 e vediamo cosa succede alle prestazione, ***quando variamo la percentuale di missing data presenti nel dataset?***

ASIA

Fissiamo il numero di dati pari a 1000 e vediamo cosa succede alle prestazione, ***quando variamo la percentuale di missing data presenti nel dataset?***

Prop	Numero missing	Numero dati	EM HARD / ground truth	EM SOFT FORCED / ground truth	EM SOFT / ground truth
0.1	97	1000	(3) 0.773	(3) 0.773	(3) 0.773
0.25	259	1000	(3) 0.795	(3) 0.795	(4) 0.795
0.4	400	1000	(3) 0.78	(3) 0.822	(4) 0.822
0.5	510	1000	(3) 0.756	(3) 0.806	(5) 0.806
0.7	699	1000	(3) 0.775	(3) 0.784	(5) 0.784
0.9	894	1000	(3) 0.803	(3) 0.802	(6) 0.803
0.99	**999**	**1000**	**(3) 0.82**	**(3) 0.818**	**(6) 0.82**

ASIA

Fissiamo il numero di dati pari a 1000 e vediamo cosa succede alle prestazione, quando variamo la percentuale di missing data presenti nel dataset?

Prop	Numero missing	Numero dati	EM HARD / ground truth	EM SOFT FORCED / ground truth	EM SOFT / ground truth
0.99	999	1000	(3) 0.82	(3) 0.818	(6) 0.82

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17
A	0	0	0	0	0	0	0	NA	0	0	0	0	0	NA	0	0	0
S	1	1	0	0	0	NA	0	1	NA	NA	0	0	1	1	0	0	0
T	NA	NA	1	NA	0	0	0	0	0	0	0	0	NA	0	NA	0	0
L	0	0	0	0	0	0	0	0	0	0	0	NA	0	0	0	0	0
B	1	0	0	1	NA	0	0	1	1	1	NA	1	1	1	0	NA	0
E	0	0	1	0	0	0	NA	0	0	0	0	0	0	0	0	0	0
X	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
D	1	0	NA	1	1	1	0	1	1	1	0	1	1	1	0	0	0

ASIA

Facendo imparare ad EM HARD troppi dati completi (PROP basso), ***la rete impara troppo bene la distribuzione delle classi di maggioranza****, i risultati sembrano essere* ***overfittati*** *e conseguentemente, le* ***prestazioni peggiorano***

ASIA

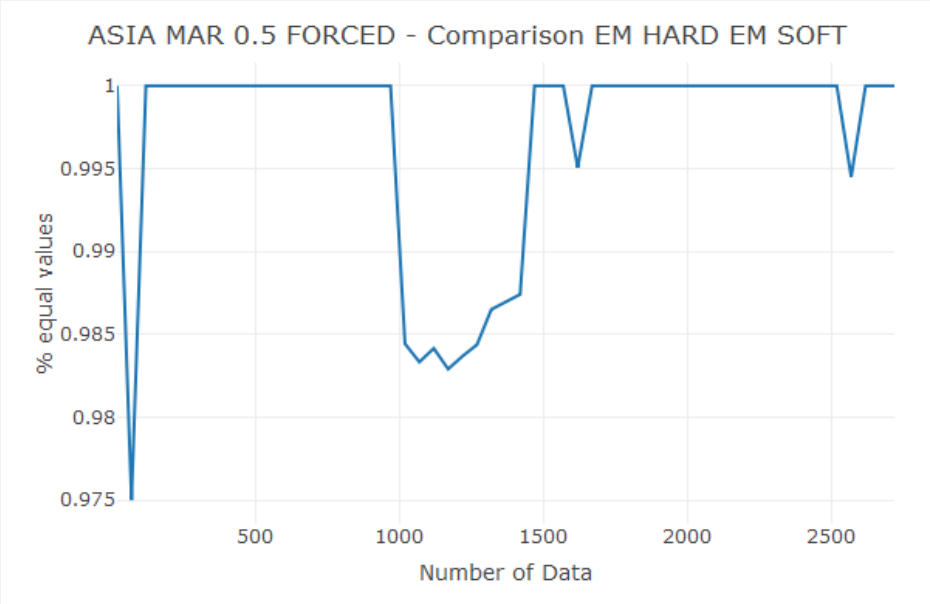
Per finire, vediamo cosa succede se generiamo dati di tipo MAR o MCAR

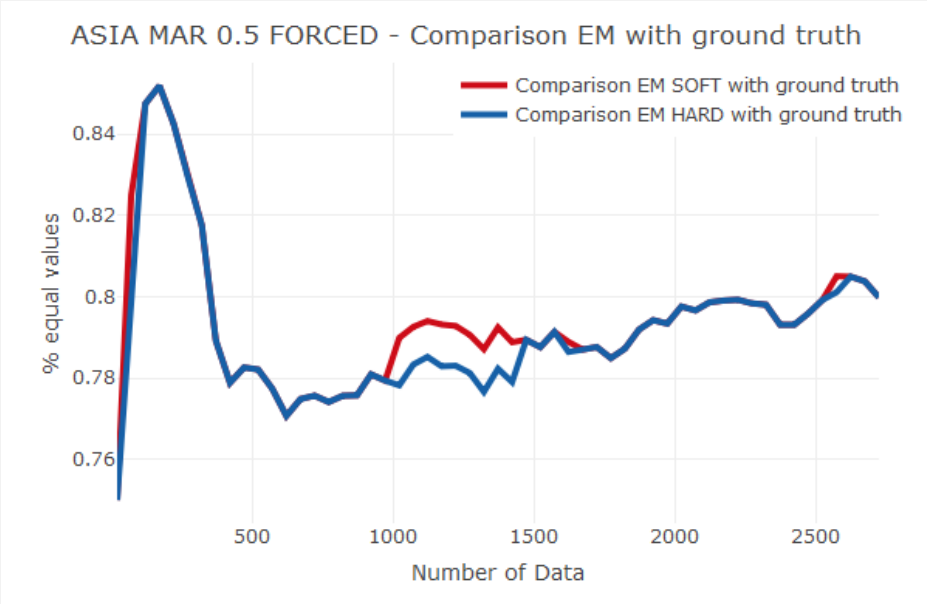
ASIA

Missing at Random (MAR)

PROP 0.5

FORCED



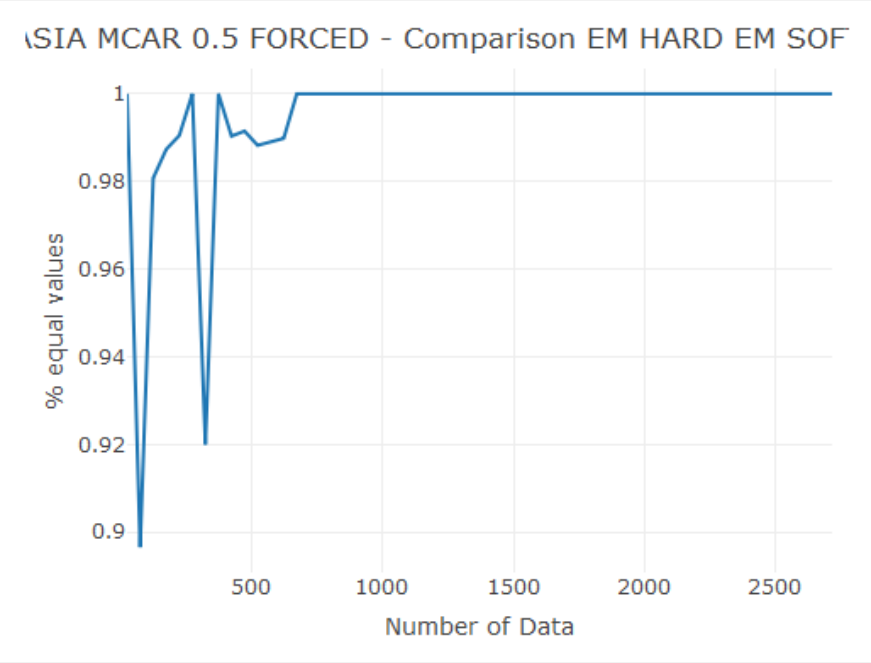


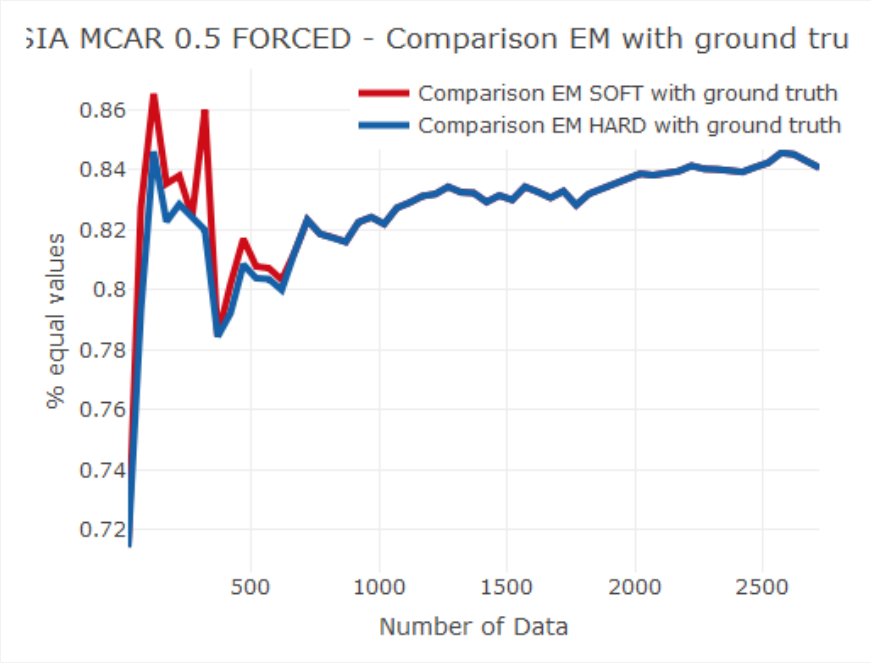
ASIA

Missing Completely at Random (MCAR)

PROP 0.5

FORCED



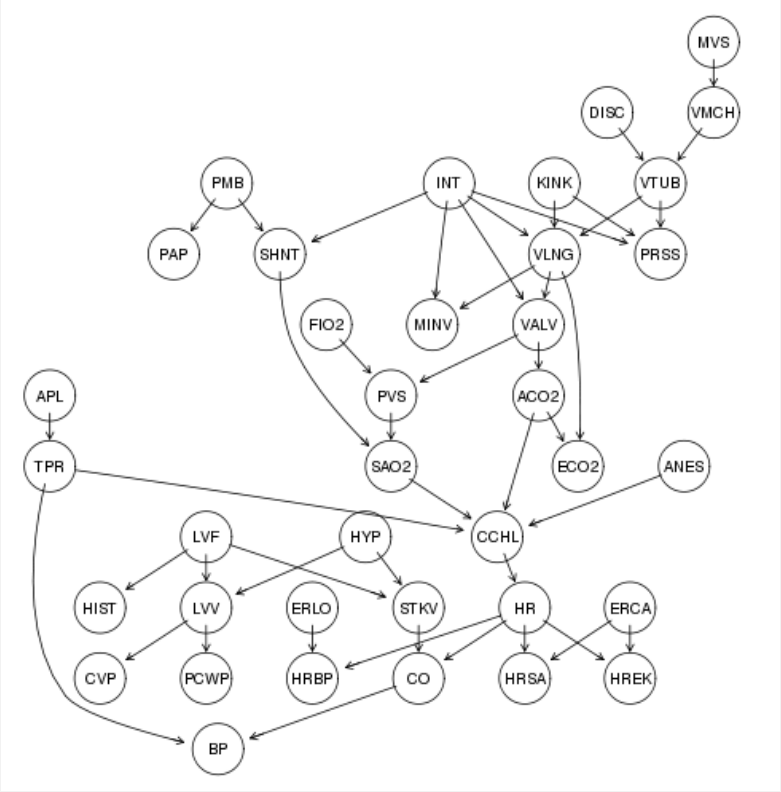


DATASET

ALARM

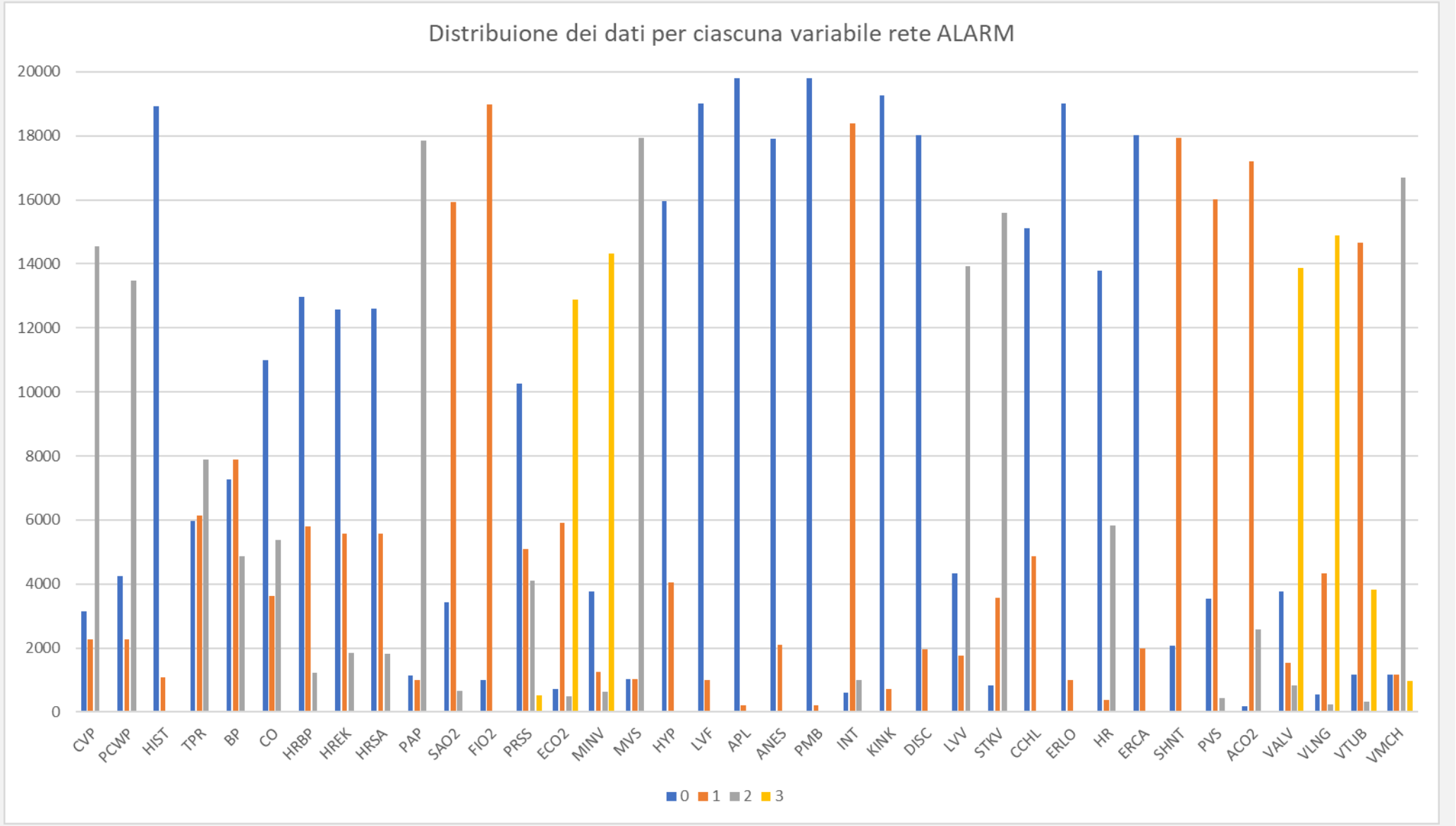
ALARM

ALARM ("*A Logical Alarm Reduction Mechanism*") è una rete Bayesiana progettata per fornire un sistema di messaggi di allarme per il monitoraggio del paziente.



Numero di nodi: 37
Numero di archi: 46
Numero di parametri: 509

ALARM

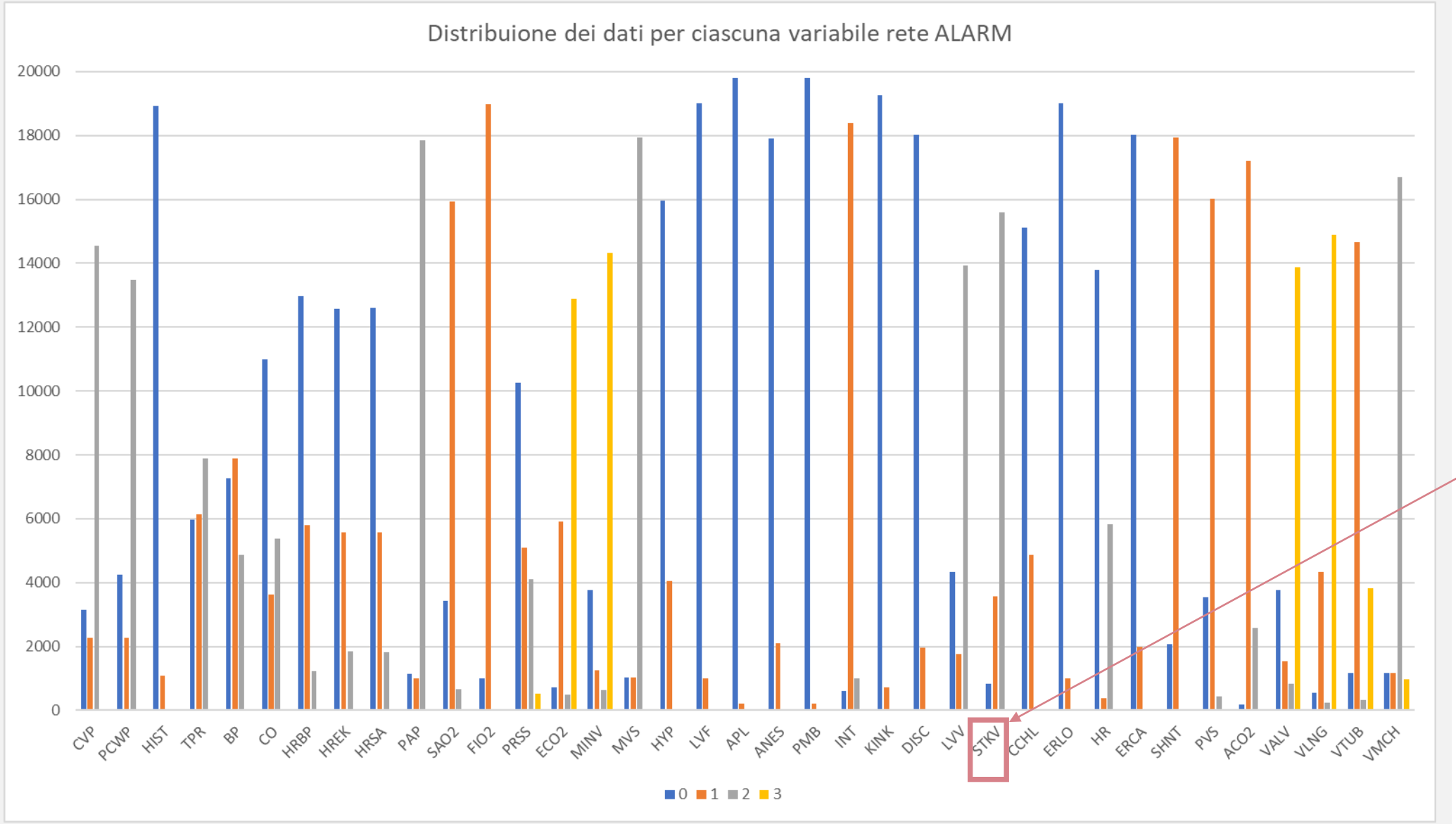


Il dataset è composto da *20.000* record e *740.000* celle

Ci sono tante variabili **sbilanciate**, ma, a differenza di ASIA, le classi di minoranza, alcune volte possono non essere rare.

Ad esempio, nella variabile STKV, la classe di maggioranza compare con una frequenza 0,78. DISC invece ha una frequenza pari allo 0,9

ALARM



Il dataset è composto da *20.000* record e *740.000* celle

Ci sono tante variabili **sbilanciate**, ma, a differenza di ASIA, le classi di minoranza, alcune volte possono non essere rare.

Ad esempio, nella variabile STKV, la classe di maggioranza compare con una frequenza 0,78. DISC invece ha una frequenza pari allo 0,9

ALARM

Generiamo dati **MNAR** attraverso il metodo `ampute` e consideriamo solo 620 dati.

Consideriamo 3 diversi contesti:

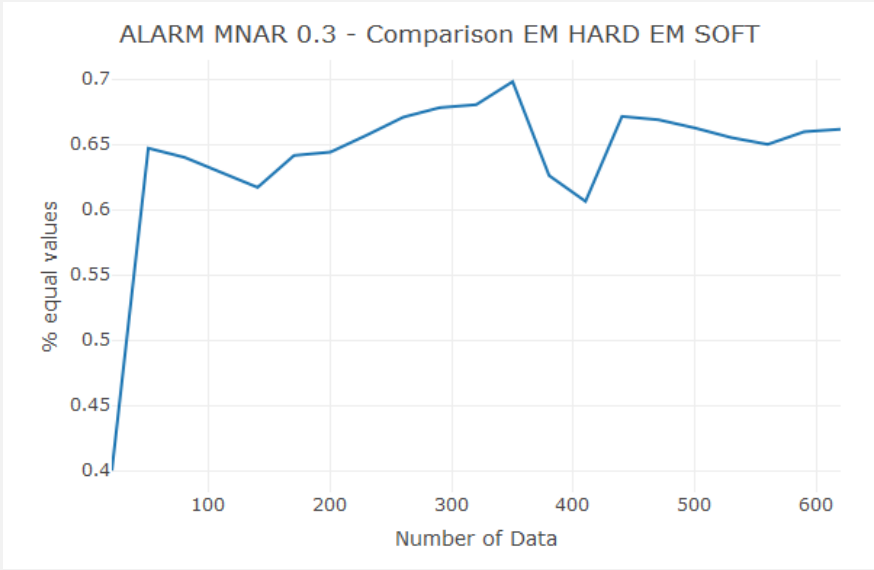
- PROP = 0.3
- PROP = 0.5
- PROP = 0.6

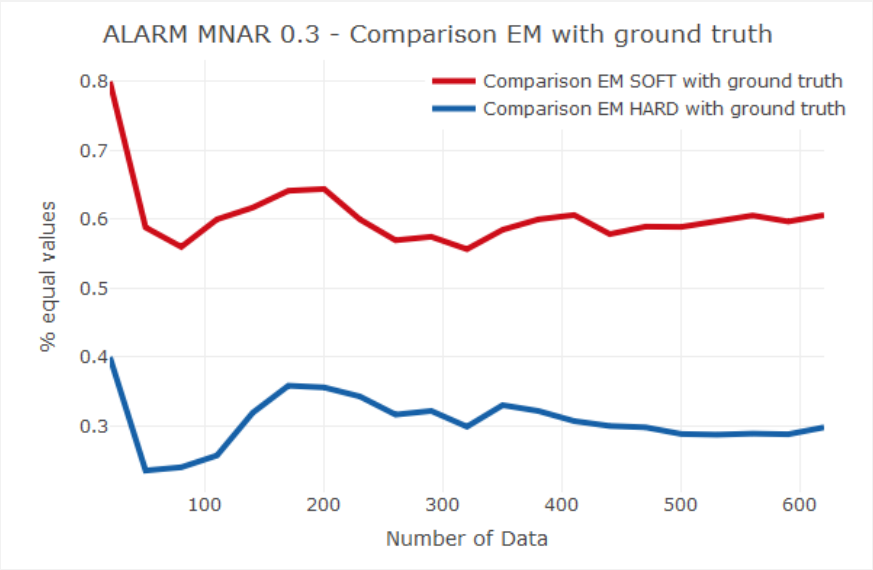
ALARM

Missing Not at Random (MNAR)

PROP 0.3

FORCED



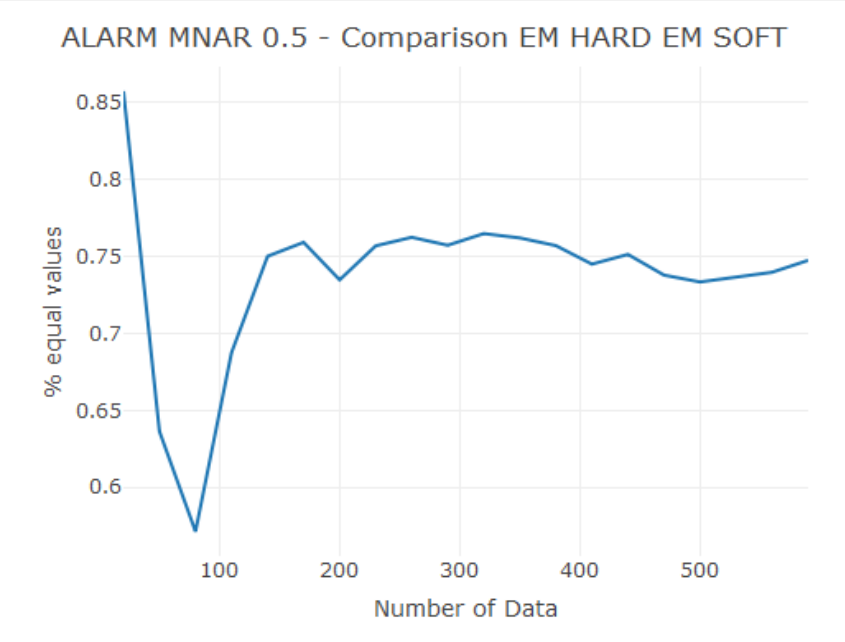


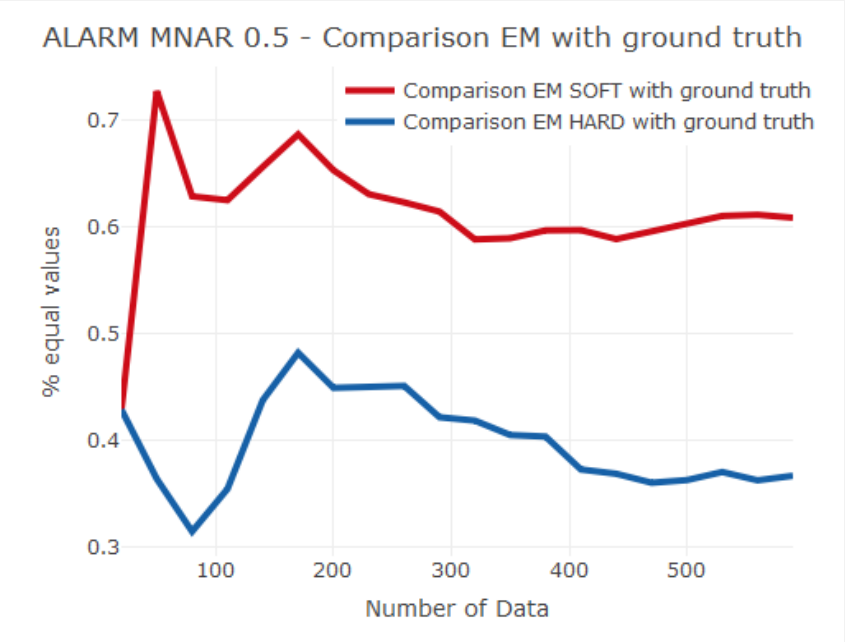
ALARM

Missing Not at Random (MNAR)

PROP 0.5

FORCED



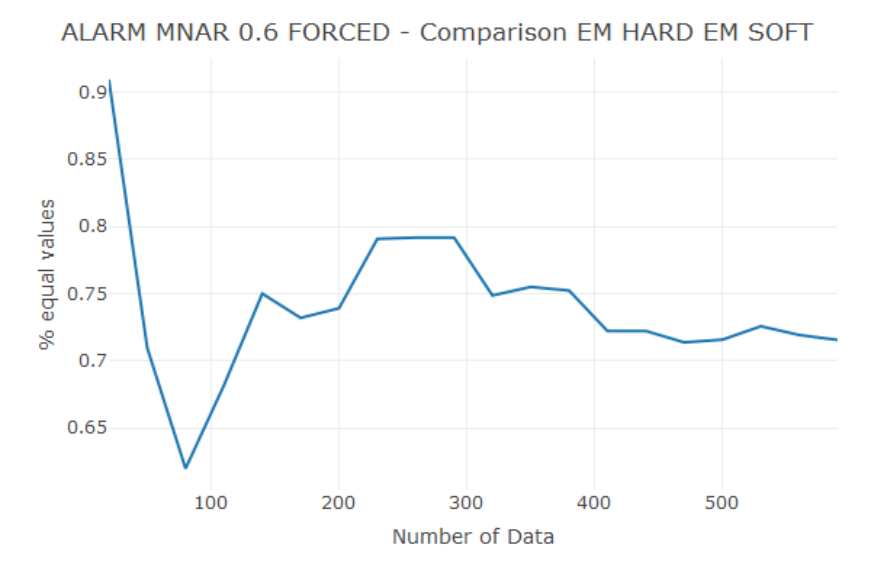


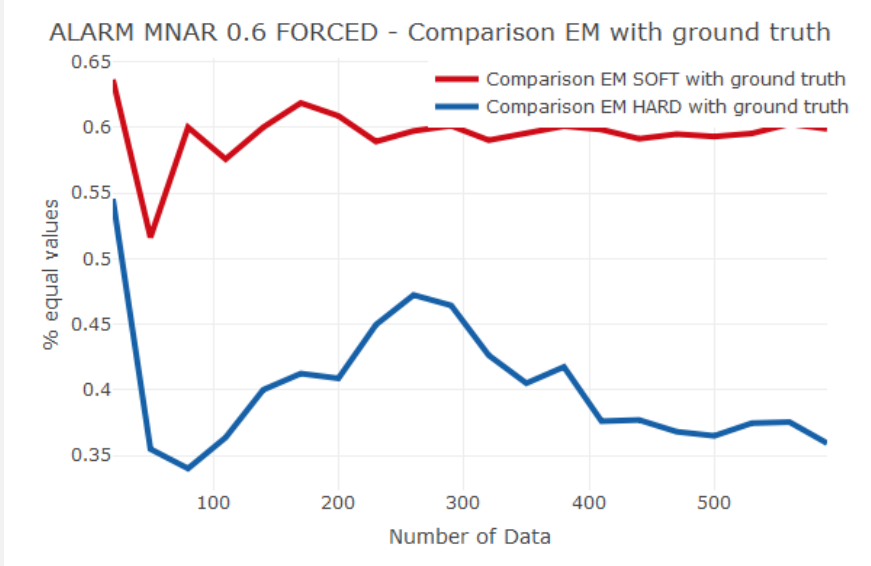
ALARM

Missing Not at Random (MNAR)

PROP 0.6

FORCED





ALARM

Fino a questo punto ci siamo limitati a simulazioni che andavano da 20 a 620 record. Cosa succede se eseguiamo EM HARD e EM SOFT su un numero decisamente maggiore di dati?

ALARM

Fino a questo punto ci siamo limitati a simulazioni che andavano da 20 dati a 620 dati. Cosa succede se eseguiamo EM HARD e EM SOFT su un numero decisamente maggiore di dati?

TIPO DI DATI	ALPHA	PROP	Numero missing	Numero dati	Iterazioni EM HARD	Iterazioni EM SOFT	Tempo EM HARD	Tempo EM SOFT	EM HARD /EM SOFT	EM HARD /Ground TRUTH	EM SOFT/Ground TRUTH
MNAR	0,05	0,3	746	2500	4	5	480 sec	660 sec	0,5844504	0,2868633	0,6260054
MNAR	0,05	0,3	746	2500	4	4 FORCED	480 sec	515 sec	0,623465	0,2868633	0,6041667
MNAR	0,05	0,3	1800	6000	4	7	2040 sec	4740 sec	0,598755	0,2840973	0,6123373
MNAR	0,05	0,3	1800	6000	4	4 (FORCED)	2040 sec	2567 sec	0,60705	0,2840973	0,6041667

ALARM

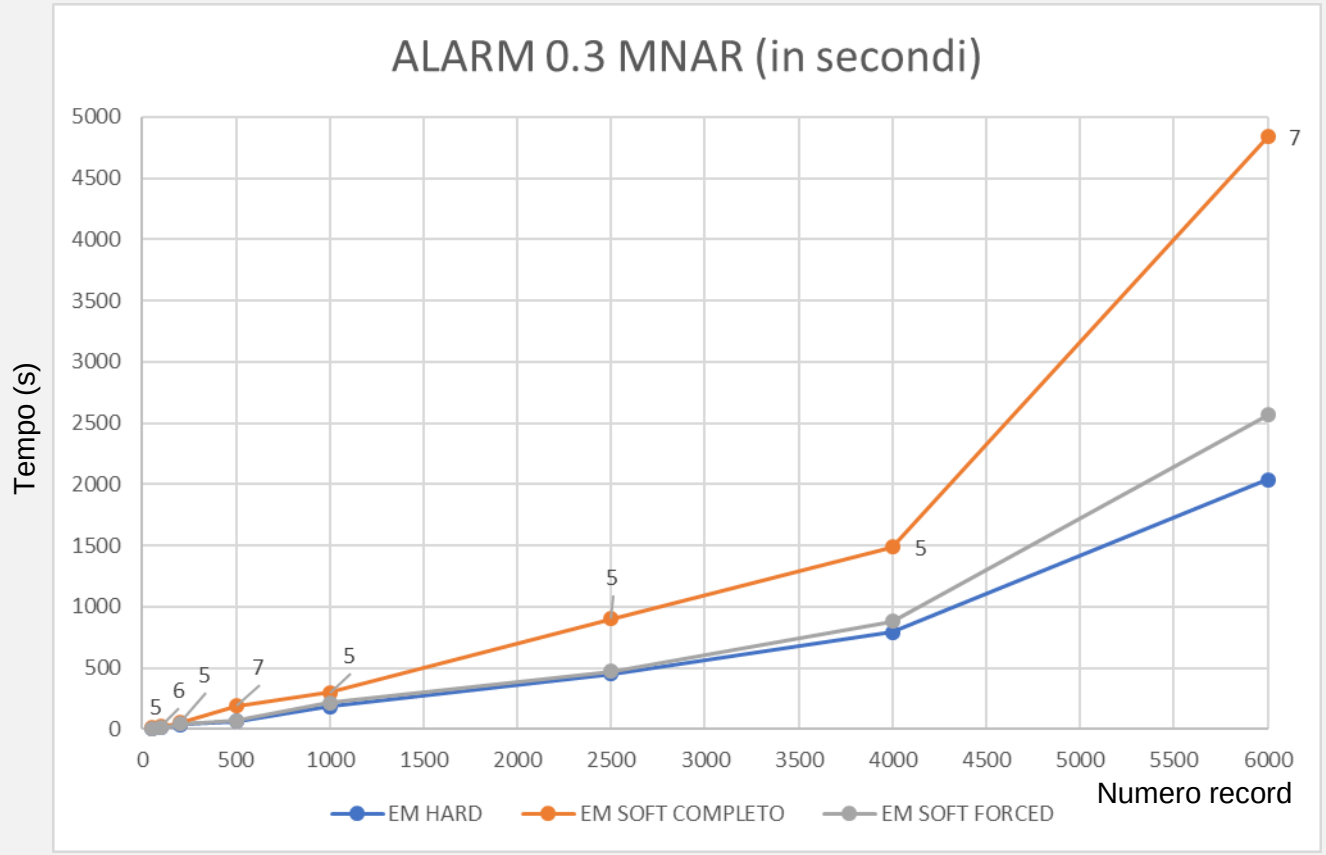
Fino a questo punto ci siamo limitati a simulazioni che andavano da 20 dati a 620 dati. Cosa succede se eseguiamo EM HARD e EM SOFT su un numero decisamente maggiore di dati?

TIPO DI DATI	ALPHA	PROP	Numero missing	Numero dati	Iterazioni EM HARD	Iterazioni EM SOFT	Tempo EM HARD	Tempo EM SOFT	EM HARD /EM SOFT	EM HARD /Ground TRUTH	EM SOFT/Ground TRUTH
MNAR	0,05	0,3	746	2500	4	5	480 sec	660 sec	0,5844504	0,2868633	0,6260054
MNAR	0,05	0,3	746	2500	4	4 FORCED	480 sec	515 sec	0,623465	0,2868633	0,6041667
MNAR	0,05	0,3	1800	6000	4	7	2040 sec	4740 sec	0,598755	0,2840973	0,6123373
MNAR	0,05	0,3	1800	6000	4	4 (FORCED)	2040 sec	2567 sec	0,60705	0,2840973	0,6041667

Guardando questi risultati, risulta che EM HARD sia nettamente inferiore e che riesca a classificare correttamente «solo» 3 dati su 10. Tuttavia, importanti considerazioni verranno presentate a breve.

ALARM

Vediamo ora come si comportano EM HARD ed EM SOFT in termini di ***tempo computazionale****.*



ALARM

Per finire, fissiamo il numero di dati pari a 500 e vediamo cosa succede alle prestazioni, variando la ***percentuale di missing data*** *presenti nel dataset.*

Prop	Numero missing	Numero dati	EM HARD / ground truth	EM SOFT FORCED / ground truth	EM SOFT / ground truth
0.1	48	500	(3) 0.245	(3) 0.623	(5) 0.623
0.25	124	500	(3) 0.317	(3) 0.575	(7) 0.575
0.4	200	500	(3) 0.333	(3) 0.631	(5) 0.631
0.5	247	500	(3) 0.363	(3) 0.604	(7) 0.604
0.7	349	500	(3) 0.424	(3) 0.61	(8) 0.614
0.9	450	500	(4) 0.501	(4) 0.6	(10) 0.62
0.99	**499**	**500**	**(3) 0.519**	**(3) 0.603**	**(8) 0.605**

ALARM

Tuttavia, osservando l'esecuzione di EM HARD, ci si trova di fronte a un comportamento non atteso.

ALARM

EM HARD: caso particolare

Fissando il numero di record a 200, con prop 0.5 e alpha 0.05**,** l'algoritmo non termina affinché non si raggiunge il numero di iterazioni massime.

Ad ogni iterazione, la differenza tra le CPT dello step precedente e le CPT dello step attuale risulta essere costante (1,440015).

Possibile motivazione: quando si ha una scarsa quantità di dati, EM HARD cambia in modo periodico, i valori dei missing data. In questo modo il meccanismo di *stopping_criteria*, definito come differenza in valore assoluto con le CPT del passo precedente, fallisce.

ALARM

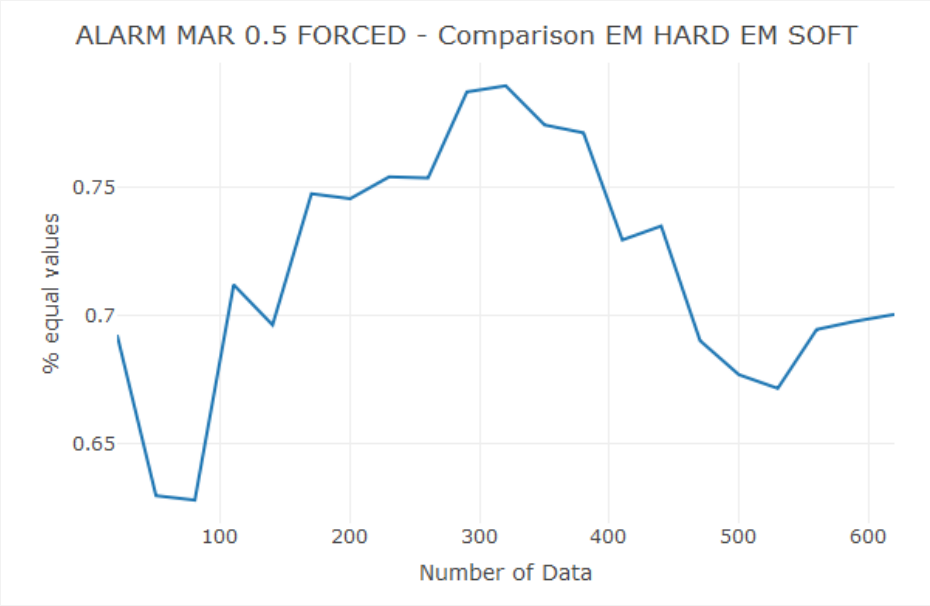
Risultati analoghi sono stati osservati su dati di tipo MAR e MCAR. Tuttavia, in questo caso, le prestazioni di EM HARD ed EM SOFT risultano essere migliori rispetto alle prestazioni con dati MNAR.

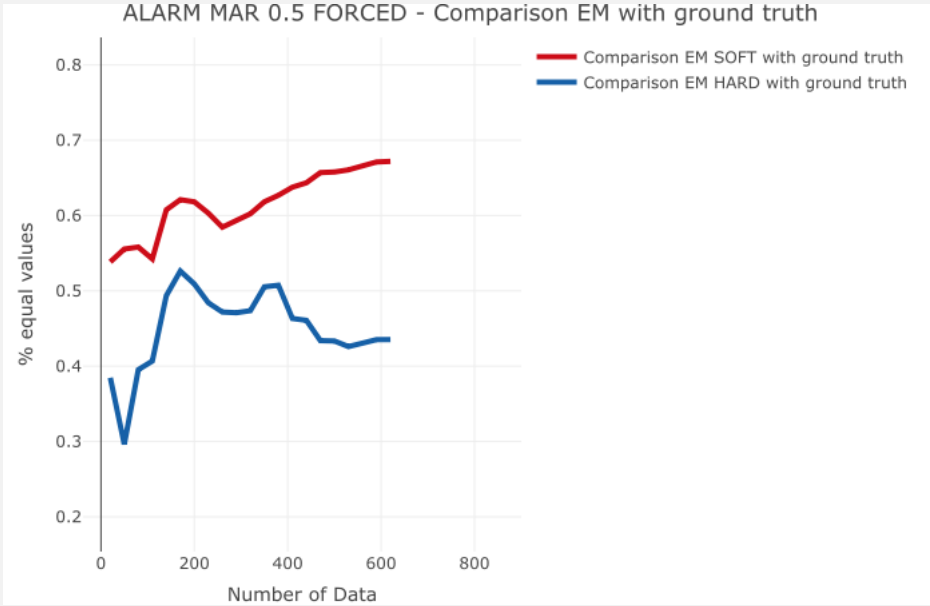
ALARM

Missing at Random (MAR)

PROP 0.5

FORCED





CONCLUSIONI

EM HARD vs EM SOFT

Problemi e Sviluppi Futuri

CONCLUSIONI

EM HARD vs EM SOFT

Le prestazioni dei due algoritmi dipendono molto dai dati in input. EM HARD soffre maggiormente quando le CPT tendono ad assegnare una **probabilità uniforme** a ciascun valore assumibile dalla variabile;

In reti come *ASIA* non c'è differenza sostanziale tra EM SOFT ed EM HARD in quanto le CPT tendono ad assegnare massima probabilità ad un valore.

CONCLUSIONI

EM HARD vs EM SOFT

La **convergenza** di EM SOFT è molto più lenta: ci vogliono molte più iterazioni rispetto ad EM HARD.
Inoltre il **tempo computazionale** richiesto è molto alto: quando EM HARD termina alla terza o quarta iterazione (sia in ASIA sia in ALARM), EM SOFT può impiegare fino a 10 iterazioni per terminare la sua computazione. In media, per arrivare all'arresto, EM SOFT ci mette **almeno il doppio del tempo** e questo può rendere la computazione complessa su dataset o reti molto grandi.

CONCLUSIONI

EM HARD vs EM SOFT

Se forziamo il **numero di iterazioni massime** di EM SOFT ad essere uguali a quelle di EM HARD, le prestazioni di EM HARD potrebbero addirittura essere migliori su alcuni tipi di rete rispetto ad EM SOFT;

Non si osservano però differenze significative in termine di prestazioni tra EM SOFT FORCED ed EM SOFT nelle reti ASIA e ALARM;

EM HARD non garantisce l'arresto dell'algoritmo considerando esclusivamente la differenza tra le CPT. E' stato osservato uno scenario dove i valori assumibili dalle variabili cambiavano in maniera periodica.

CONCLUSIONI

EM HARD vs EM SOFT

Quando abbiamo a disposizione una piccola percentuale di missing data, un algoritmo che effettua tante iterazioni può essere molto efficiente. Al contrario, se il dataset è molto grande e la rete è molto complessa, gli algoritmi impiegano una quantità di tempo troppo alta.

Si è notato che EM HARD raggiunge prestazioni migliori **quanto è minore il numero di dati completi** presenti nel dataset.

CONCLUSIONI

EM HARD vs EM SOFT

Per concludere, la scelta di quale algoritmo utilizzare tra EM HARD o EM SOFT dipende sempre dal contesto e non è possibile stabilire a priori qual è l'algoritmo migliore. Tuttavia, di seguito, vengono riassunte alcune osservazioni:

- Se il dataset è costituito da pochi dati, **EM SOFT** è la soluzione preferibile in quanto l'algoritmo impara su tutti i possibili assegnamenti dei valori alle variabili;

- Se il dataset è costituito da tanti dati ma la proporzione dei dati missing è bassa, **EM SOFT FORCED** è la soluzione preferibile;

- Se il dataset è costituito da tanti dati e la proporzione dei missing data è molto alta, **EM HARD** è la soluzione preferibile, considerando anche i tempi computazionali;

- Se la rete presenta **distribuzioni di probabilità uniformi** (massima incertezza), le prestazioni di EM HARD ne risentono molto ed EM SOFT risulta essere consigliabile.

REPOSITORY

Presentazione repository GitHub