UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

MOE: Mixture Of Experts

Analisi della letteratura e delle Hierarchical MOE

Modellazione Statistica - Data Science

Ivan Diliso - 761053

Indice

1 Mixture of Experts

1.1 Introduzione

Le Mixture of Experts (MOE) è una tecnica di enseble learning di tipologia multi expert (diversi learner che lavorano in parallelo) con approccio locale (learner selection) applicabile in constesti di apprendimento supervisionato, come classificazione e regressione, e non supervisionato come clustering. Nascono nel campo delle reti neurali nell'ambito dei comining classifiers e ensemble of weak learners. I modelli MOE utilizzano la metodologia dividi et impera per addestrare una serie di modelli parametrici e unire i loro ouput per fornire il risultato finale, questa tipologia di modelli permette di dividere un task complesso in sottotask, addestrando i vari modelli dell'esempble su un sottotask diverso. Nella terminologia delle MOE ogni modello addestrato è chiamato **esperto**. La caratteristica fondamentale che distingue le MOE è l'utilizzo di un modello ad hoc chiamato **gating** che permette di unire le soluzioni degli esperti ed assegnare ad ognuno di essi il peso che la loro soluzione avrà nel comporre la soluzione finale.

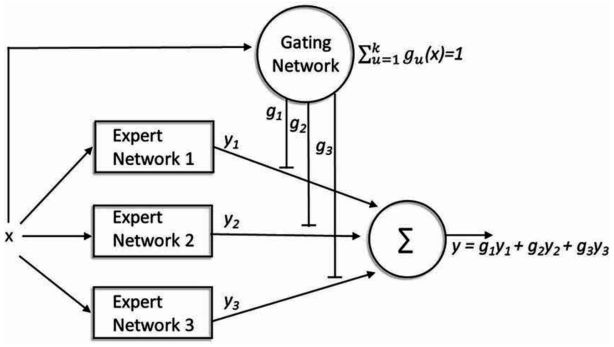


Figura 1: MOE con 3 esperti

A differenze di altre metodologie ensemble che producono esperti unbiased con stime di errore non correlate, MOE produce esperti biased con stime correlate negativamente. Con questa tipologia di modelli è necessaria una conoscenza pregressa di una divisinoe dei dati, è necessario quindi che il dataset sia divisibile.

1.2 Experts

Gli esperti sono i vari modelli addestrati nell'esemble MOE, ogni esperto andrà a specializzasi nel risolvere uno specifico sottoproblema, questo si concretizza nello specializzarsi su specifiche porzioni del feature space. I sottotask creati infatti possono essere sovrapponibili (più esperti condividono alcune feature) non sovrapponibili (ogni esperto viene addestrato su un feature space diverso). Ogni esperto quindi contribuisce in modo diverso nella creazione dell'output finale. Nelle MOE infatti non è necessario rendere i vari learner dell'esemble diverse in quanto naturalmente questi lavoreranno su task diversi. Il problema da risolvere è infatti trovare una divisione naturale dei dati, questo problema è risolto dall'utilizzo del modello di gating. Per il partizionamento del feature space, possono essere utilizzate due metodologie:

- Partizionamento **inplicito**: Feature space viene diviso implicitamente in sottospazi tramite una funzione di errore. L'utilizzo dell'apprendimento tramite backpropagation e il partizionamento dei dati fornito dalla funzione di gating permeette agli esperti di specializzarsi in diversi sottospazi del feature space. In questo caso di parla di approccio **MILE** (Mixture of the Implicitly Localised Experts)
- Partizionamento **esplicito**: Si utilizza un algoritmo di clustering per il partizionamento dei dati, questi vengono poi assegnati ad un esperto. In questo caso si parla di un approccio **MELE** (Mixture of Expliticly Localised Experts)

Un esperto può essere qualsiasi modello di apprendimento, nella formulazione originale gli esperti prendono la forma di modelli di logistic regression, ma possono essere anche implementati come percettroni, multi layer perceptron, gaussian mixture, reti neurali profonde (in questo caso di parla di Deep Mixture of Experts) etc. etc.

1.3 Gating

Il modello di gating può essere di diverse tipologie e assolve una funzione specifica in base alla sua tipologia. Nella formulazione originale il modello di gating è un modello che, dato in input un elemento del dataset, fornisce in output una distribuzione di probabilità sull'insieme di esperti, questa distribuzione può essere utilizzata in due modi:

- **Pooling**: L'esperto a cui è associata la probabilità più alta verrà selezionato come il modello per la predizione dei dati.
- **Combining**: Le soluzioni di tutti gli esperti vengono combinate pesate con la probabilità fornita dalla funzione di gating, ogni esperto quindi contribuisce proporzionalmente al peso generato dal gating.

Questo approccio associa quindi un diverso peso ad ogni esperto, questa tecnica può essere vista come una forma di voting dei modelli ensemble, dove però la capacità di voto può cambiare al variare dell'input. Dato che la distribuzione è appresa dinamicamente dal modello di gating sulla base dell'input questa apprende che input assegnare ad ogni esperto effettuando quindi un partizionamento dei dati. Questo partizionamento è definito **hard** se si utilizza una tecnica di pooling, **soft** se si utilizza il combining.

1.4 Hierarchical Mixture of Experts

Le Hierrarchical Mixture of Experts (HMOE) sono una estensione dei modelli MOE in cui rimpiazzo ogni esperto con un sistema completo MOE in modo ricorsivo creando una struttura ad albero in cui i nodi interni saranno le varie funzioni di gating sviluppate in modo gerarchico, e i nodi foglia conterranno gli esperti. Può essere interpretato come un albero di decisione in cui i nodi interni (gating) fungono da nodi di decisione e la decisione finale è un singolo nodo foglia (in caso di pooling) o una combinazione pesata dei nodi foglia (in caso di combining). Se nei modelli MOE un parametro importante per l'ottimizzazione del modello è il numero di esperti da utilizzare in questo caso il parametro è la profondità della ricorsione, che determinerà il numero di esperti utilizzati.

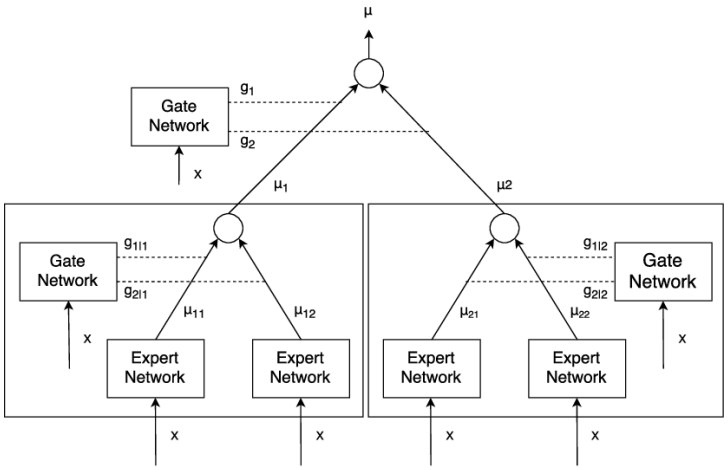


Figura 2: HMOE con con profondità 2

Questa tipologia di albero, con gating di tipo combining viene definita **soft decision tree** in quanto a differenze degli alberi di decisione hard, vengono utilizzate le distribuzioni di probabilità generate dai nodi gating per esplorare tutti i possibili percorsi dell'albero, per poi effettuare una somma pesata a livello di foglie, dove ogni peso è uguale al prodotto dei valor idi gating di ogni path per arrivare alla foglia. A differenza dei modelli CART (Classification And Regression Trees) dove ogni nodo contiene un valore costante per la decisione, nei modelli HMO ogni nodo implementa un modello lineare (o di regressione logistica). L'uso di questa formulazione soft permette di catturare situazioni in cui la transizione da una risposta alta ad una bassa è graduale.

2 Apprendimento e backpropagation

2.1 Modello MOE

Sia $x \in \mathbb{R}^n$ vettore di input, sia T il numero di esperti modello, $h_1, \ldots, h_T$ gli esperti del modello e y variabile target. Dati W_i parametri dell'i-esimo esperto, l'obiettivo del modello è approssimare la distribuzione di y sulla base dei dati di training

$$h_i(y|x; W_i)$$

La funzione di gating produce un set di coefficenti the pesano il contributo degli esperti, sia v_i vettore dei pesi della funzione di gating relativa all'i-esimo esperto e sia α insieme dei paramatri dell modello di gating (insieme dei pesi relativi ad ogni espeto). Essendo una distribuzione di probabilità la somma dei coefficenti su tutti gli esperti dovrà essere uguale a 1. Definiamo quindi il set di coefficenti generati dalla funzione di gating, sull'i-esimo esperto, dato l'insieme di parametri α come:

$$\pi_i(x; \alpha) : \sum_{i=1}^{T} \pi_i(x; \alpha) = 1$$

Sulla base di queste probabilità partizioniamo lo spazio di input, diverse partizioni appartengono a diversi esperti. L'output del modello, dato ψ insieme dei parametri degli esperti e del modello di gating, sarà quindi:

$$H(y|x; \psi) = \sum_{i=1}^{T} \pi_i(x; \alpha) \cdot h_i(y|x; W_i)$$

Nella fase di training il valore $\pi_i(x; \alpha)$ indica la probabilità che l'istanza x appaia nel traning set dell'i-esimo esperto. Mentre nella fase di testing definisce il contributo che il modello h_i fornisce alla predizione finale. L'output della funzione di gating può essere espresso tramite una softmax:

$$\pi_i(x; \alpha) = \frac{e^{v_i} x}{\sum_{l=1}^{k} e^{v_l} x}$$

usata sia per classificazione che per regressione.

2.2 Modello HMOE

Seguendo il formalismo definito in precedenza definiamo un modello HMOE di profondità 2. Definito I numero di nodi connessi al nodo gating al livello di radice, J_i numero di nodi connessi all'i-esimo nodo gating, π_i ougput del nodo gating al livello di radice e $\pi_{j|i}$ output dell j-esimo nodo gating connesso all'i-esimo nodo gating, l'ouput del modello HMOE sarà:

$$H(y|x; \psi) = \sum_{i=1}^{I} \pi_i(x; \alpha_{\pi_i}) \cdot \sum_{j=1}^{J_i} \pi_{j|i}(x; \alpha_{\pi_{j|i}}) h_{ij}(y|x; W_{ij})$$

Nella fase di apprendimento e testing i dati sono dati in input agli esperti che producono dei vettori di output che procedono nell'albero verso alto (in direzione opposta rispetto ai modelli CART), vengono moltiplicati tra loro e sommati seguendo i vari livelli dell'albero.

2.3 Apprendimento

Gating network alloca dati di training a uno o più esperti e se l'output è incorretto il cambiamento dei pesi e localizzato su questo esperto. Locale in quasnto i pesi di un esperto sono disaccoppiati dai pesi di un altr oesperto.

Può avvenire tramite:

- GRADIENT DESCENT: Particolarmente utile con i mixuture of multi layer perceptron experts. Addestramento utilizzando questa funzione tende ad assegnare un dato di training ad ogni esperto
- EXPECTATION MAXIMIZATION: Metodi ME cercano di risolvere due task, dato un esperto trovare la funzione di gating ottimale e data la funzinoe di gating addestrare ogni esperto a massimizzare le performance sulla distribuzione assegnata dalla funzione di gating, questo rende naturale l'utilizzo di un algoritmo di expectation maximization

2.4 Funzioni di errore

$$E = \|y - \sum_j g_j O_j\|^2$$

I pesi di ogni esperto sono cosi aggiornati sulla base di un errore ensemble totale che sulla base dell'errore dello specifico esperto. Questo permette un alto livello di cooperazione e tende a sfruttare quasi tutti gli esperti del modello (nessun esperto non contribuisce al problema) In questa funzione di errore si assume che che l'output del sistema sia una combinazione lineare degli output degli esperti locali, con il gating che determina la proporzione. Strong coupling dei pesi.

$$E = \sum_j g_j \|y - O_j\|^2$$

Pesi aggiornati su errori singoli, non assicura la localizzazione degli esperti.

2.5 Errore con gaussian mixture

Una misura di errore che tiene conto di entrambi i fattori è basata sulla negative log probability di generare l'output vector desiderato, se si assume una mixture di modelli gaussiani con $\sum$ matrice di covarianza

$$E_{ME} = -log \sum_j g_j e^{-\frac{1}{2}(y-O_j)^T \sum^{-1} (y-O_j)}$$

L'apprendimento di ogni esperto avviene sull'errore individuale, ma l'aggiornamento dei pesi per ogni esperto è propozionale all suo rateo di errore sull'errore totale. Questo permette la localizzazione degli esperti nei sotto spazio delel feature corrispondente

2.6 Errore con MLP-Experts

Ogni esperto è un MLP con un hidden layer che produce un output O_j in funzione dell'inpnt con funzione di attivazione sigmoidale. Apprendimento con backpropagation massimizzando la log likelihood dei dati i parametri.

2.7 Model selection

Ottimizzazione di iperparamtrei, depth e connessioni dell'albero. Simile alla model selection applicata ad alberi, modificata la funzione di valutazione di un branch dell'albero.

- Modelli growing: aggiungo layer all'albero e determno la profondità e numero di esperti
- Pruning modelli: Riduzione dei requirement computazionali. Parametri costanti ma considero le path più probabili. Pruno i banch meno usati

3 MOE e HMOE in R

MEclusternet, flexmix, mixreg, mixtools, flexCWM, meteorist

3.1 mixtools: Tools for Analyzing Finite Mixture Models

3.2 MoEClust

La libreria MoEClust(Gaussian Parsimonious Clustering Models with Gating and Expert Network Covariates and a Noise Component) permette di addestrare mixture of expert gaussiane finite tramite algoritmo di ottimizzazione expectation maximization, l'utilizzo principale della libreria è il clustering.

4 Clutering asteroidi pericolosi che orbitano la terra

4.1 Dataset

- Neo Reference ID e Name
- Absolute magnitude
- Est Dia: Stima del diametro dell'asteroide, vengono fornite le misure di massimo e minimo in KM, M, Miles e Feet. Feature ridondanti mantengo solo la stima massima e minima dell'asteroide in KM.
- Approach date: In format di data e epoch, vengono rimosse in quanto non utili al clustering
- Velocità relativa: In km/h, in km/s e in miles/h: Viene mantenuta solo la velocità in km/s
- Distanza mancante: In unità astronomica, kilometri, lunare e miglia. Viene mantanuta la distanza astronomicaz
- Pianeta orbitante: Contiene il valore terra per tutti i dati, id dell'orbita, giorno di verifica dell'orbita e incertezza dell'orbita e intersezione minima con le orbite
- Jupiter tisserand invariant: Descrive la dinamica orbitale dell'asteroide in ralazione a Jupyter, fornisce una misura del grado di stabilità dell'orbita, valori elevati indicano orbite stabili, basse instabili e soggette a perturbazioni.
- epoch osculation
- eccentricity
- semi major axis
- inclination
- asc node longitude
- orbital period
- perihelion distance
- periohelion arg
- aphelion dist
- perihelion time
- mean nomaly
- mean motion
- equinox
- hazadous

4.2 Rimozione feature

```
eval = FALSE
library(readr)
library(ggplot2)
library(gpairs)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

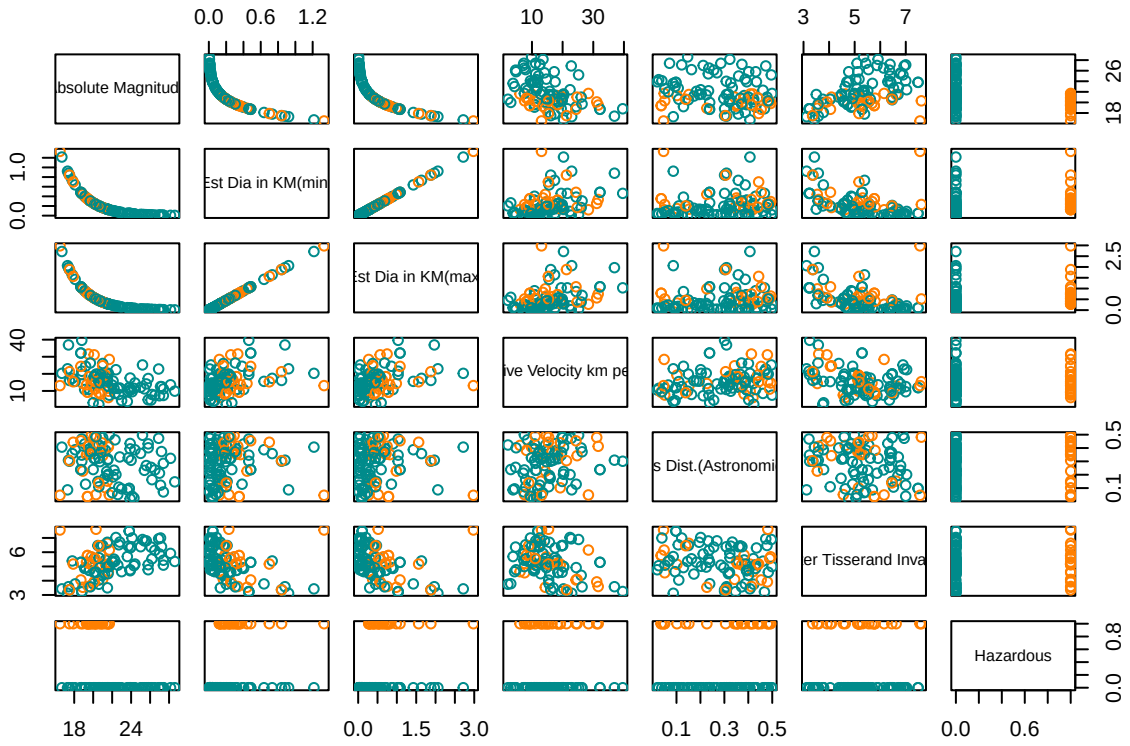
```
nasa <- read_csv("data/nasa.csv")
```

```
## Rows: 4687 Columns: 40
```

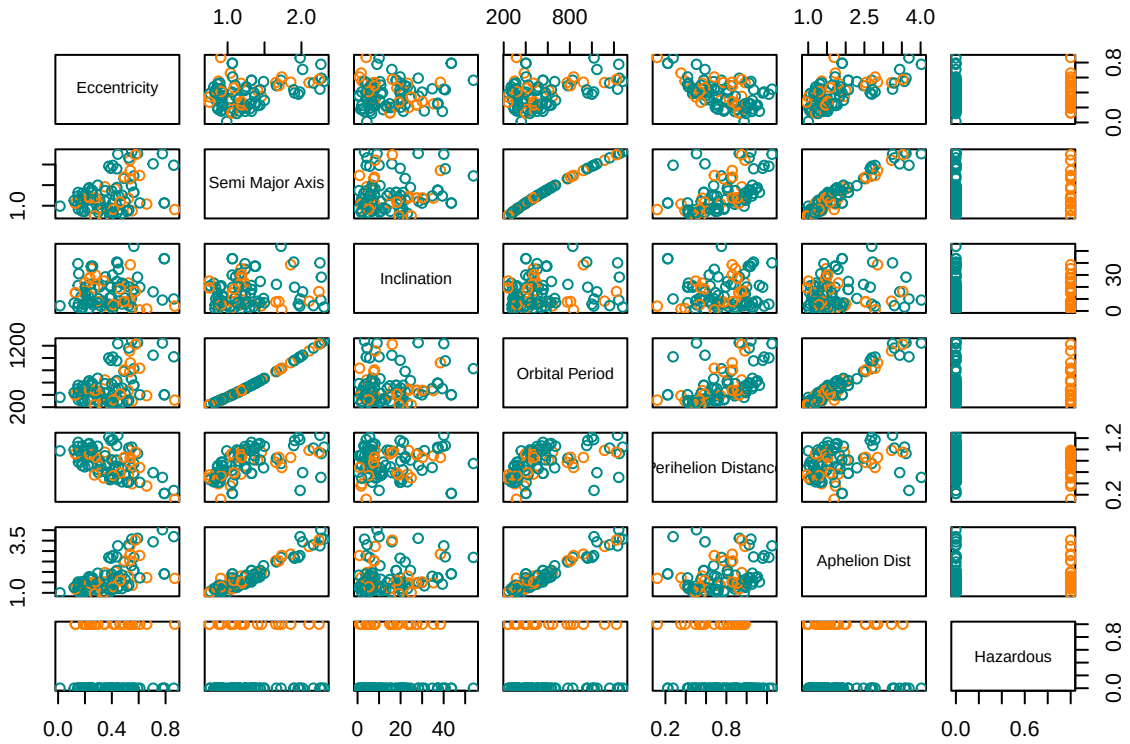
```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): Orbiting Body, Equinox
## dbl  (35): Neo Reference ID, Name, Absolute Magnitude, Est Dia in KM(min), E...
## lgl   (1): Hazardous
## dttm  (1): Orbit Determination Date
## date  (1): Close Approach Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dataset <- nasa[1:100, - (c((1:2), (6:11), (12:13), (15:16), (18:25), 27, 31,34, (36:39))), drop=FALSE]
dataset$Hazardous <- as.numeric(as.logical(dataset$Hazardous))
```

```
eval=FALSE
cols <- character(nrow(dataset))
cols[dataset$Hazardous == 0] <- "cyan4"
cols[dataset$Hazardous == 1] <- "darkorange1"

plot(dataset[, c(1:6, 13)], col=cols)
```

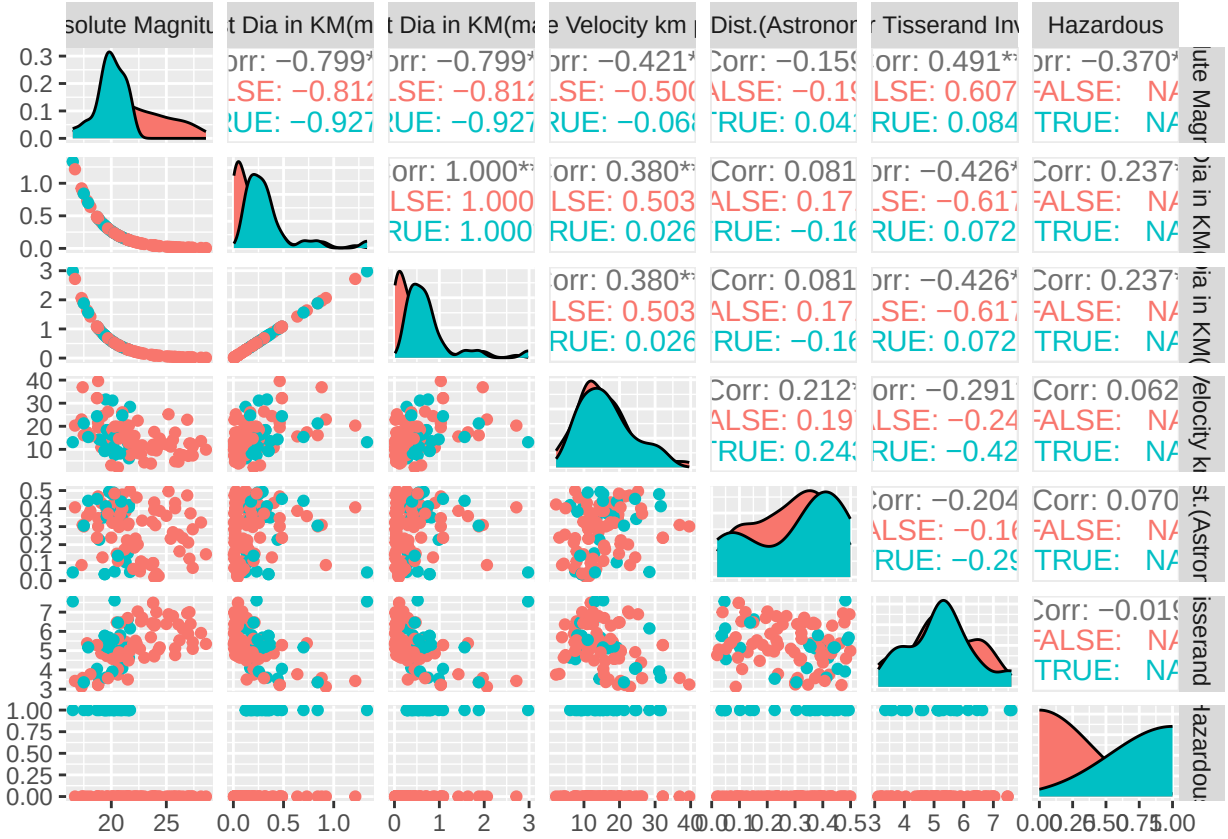



```
plot(dataset[, c(7:12, 13)], col=cols)
```




```
eval = FALSE
g1 <- ggpairs(data=dataset[, c(1:6, 13)], mapping=ggplot2::aes(colour = as.logical(Hazardous)),lower=lis

g1
```

```
## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero

## Warning in cor(x, y): the standard deviation is zero
```




```
eval=FALSE
library(MoEClust)
```

```
##
```

```
## Package 'MoEClust' version 1.5.1.
```

```
## Type '?MoEClust' to see a brief guide to how to use this R package.
## Type 'citation("MoEClust")' for citing the package in publications.
## Type 'MoE_news()' to see new features recent changes and bug fixes.
```

```
gating <- dataset$Hazardous
mod1 <- MoE_stepwise(dataset, gating)
```

```
## Warning: Potentially spurious solutions with positive log-densities were chosen at one or more steps
```

```
mod1
```

```
## ------------------------------------------------------------------------
## Comparison of Gaussian Parsimonious Clustering Models with Covariates
## Data: dataset
## Ranking Criterion: BIC
## Optimal Only: TRUE
## ------------------------------------------------------------------------
##
##  rank MoENames modelNames G  df iters       bic       icl      aic   loglik
##     1   Step_3        EEE 2 144     3 15702.445 15697.585 16077.59 8182.795
##     2   Step_2        VVV 2 208     4  8920.739  8920.739 9462.614 4939.307
##     3   Step_1        EEE 1 104     1  8571.343  8571.343 8842.281 4525.141
##  posidens gating  expert algo equalPro
##      TRUE   None ~gating   EM    FALSE
##      TRUE   None    None   EM     TRUE
##      TRUE   None    None   EM
```

```
(summ <- summary(mod1, classification=TRUE, parameters=FALSE, networks=TRUE))
```

```
## Warning: Solution contains positive log-densities and may be spurious
```

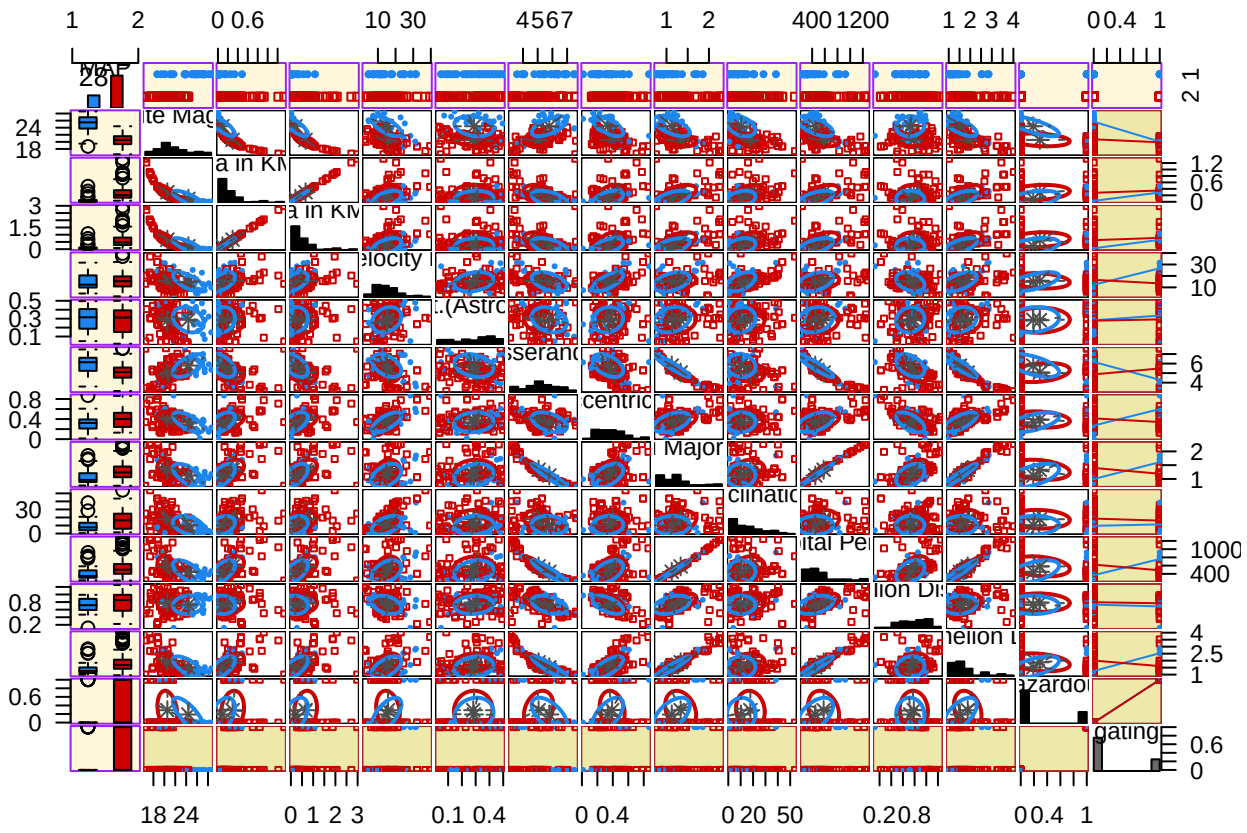
```
## ------------------------------------------------------
## Gaussian Parsimonious Clustering Model with Covariates
## Data: dataset
## ------------------------------------------------------
##
## MoEClust: EEE (ellipsoidal, equal volume, shape and orientation), with 2 components
##
## Gating Network Covariates:  None
## Expert Network Covariates:  ~gating
## Equal Mixing Proportions:   FALSE
## Noise Component:            FALSE
##
##  log.likelihood   n  d  df iters      BIC      ICL      AIC Algo
##        8182.795 100 13 144     3 15702.44 15697.58 16077.59   EM
##
## Clustering table :
##  1  2
## 28 72
```

```
## No gating network to display
```

```
## Expert Network :
##
## Cluster1 :
##
```

```
## Coefficients:
##             Absolute Magnitude Est Dia in KM(min) Est Dia in KM(max)
## (Intercept)          25.664895         0.02760841         0.06173429
## gating               -5.664829         0.27761203         0.62075938
##             Relative Velocity km per sec Miss Dist.(Astronomical)
## (Intercept)                     12.24090               0.27753040
## gating                          14.10105               0.05567917
##             Jupiter Tisserand Invariant Eccentricity Semi Major Axis
## (Intercept)                    6.184871    0.2721834       0.9948241
## gating                        -1.909611    0.3250575       0.6381520
##             Inclination Orbital Period Perihelion Distance Aphelion Dist
## (Intercept)    9.519052       366.1098          0.73024466      1.259404
## gating         1.959145       411.8266         -0.03298194      1.309286
##                 Hazardous
## (Intercept) -4.074582e-17
## gating       1.000000e+00
##
## Cluster2 :
##
## Coefficients:
##             Absolute Magnitude Est Dia in KM(min) Est Dia in KM(max)
## (Intercept)         20.7108217         0.27956738          0.6251317
## gating              -0.8148475         0.06603731          0.1476639
##             Relative Velocity km per sec Miss Dist.(Astronomical)
## (Intercept)                     16.648015               0.27865443
## gating                          -3.282121               0.01285796
##             Jupiter Tisserand Invariant Eccentricity Semi Major Axis
## (Intercept)                    4.8578727   0.41366471       1.4032384
## gating                         0.6561724  -0.07580872      -0.2279046
##             Inclination Orbital Period Perihelion Distance Aphelion Dist
## (Intercept)   18.736441       626.9105          0.79713965     2.0093371
## gating        -2.587582      -149.5408         -0.03065338    -0.4251558
##                 Hazardous
## (Intercept) -5.296435e-17
## gating       1.000000e+00
##
## Formula: ~gating
```

```
eval = FALSE
plot(mod1, what="gpairs", jitter=FALSE)
```



```
eval= FALSE
library(MoEClust)
library(caret)
```

```
## Loading required package: lattice
```

```
train_ind <- sample(1:nrow(dataset), 70)
train_set <- dataset[train_ind, ]
test_set <- dataset[-train_ind,, drop=FALSE]
haz = train_set$Hazardous

model <- MoE_clust(data = train_set, G=5, equalPro = FALSE, gating = ~ Hazardous, network.data = train_
```

```
## Warning: Gating covariates found in response data!
```

```
## Warning: Optimal model contains positive log-densities; consider setting 'posidens' to FALSE
```

```
a <- predict(model, test_set[,], use.y = FALSE)[1:9]$MAPy[,13]

a <- as.factor(as.numeric((a > 0.4)))
b <- as.factor(test_set$Hazardous)



example <- confusionMatrix(data=a, reference = b)
```

```
## Warning in confusionMatrix.default(data = a, reference = b): Levels are not in
## the same order for reference and data. Refactoring data to match.
```

```
example
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  6
##          1  0  0
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.8
##     P-Value [Acc > NIR] : 0.60697
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 1.0
##             Specificity : 0.0
##          Pos Pred Value : 0.8
##          Neg Pred Value : NaN
##              Prevalence : 0.8
##          Detection Rate : 0.8
##    Detection Prevalence : 1.0
##       Balanced Accuracy : 0.5
##
##        'Positive' Class : 0
##
```

Riferimenti bibliografici

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[2] Alpaydin, E. (2020). Introduction to machine learning. MIT press.

[3] Zhou, Z. H. (2012). Ensemble methods: foundations and algorithms. CRC press.

[4] Zhang, C., & Ma, Y. (Eds.). (2012). Ensemble machine learning: methods and applications. Springer Science & Business Media.

[5] Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: springer.

[6] Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: a literature survey. The Artificial Intelligence Review, 42(2), 275.

[7] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. Neural computation, 3(1), 79-87.

[8] Yuksel, S. E., Wilson, J. N., & Gader, P. D. (2012). Twenty years of mixture of experts. IEEE transactions on neural networks and learning systems, 23(8), 1177-1193.

[9] Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. Neural computation, 6(2), 181-214.

[10] Gormley, I. C., & Frühwirth-Schnatter, S. (2019). Mixture of experts models. In Handbook of mixture analysis (pp. 271-307). Chapman and Hall/CRC.