Linux Day

26 Ottobre 2019

linuxday.thefreecircle.org/2019/

Riconoscimento delle immagini
Con R e l'utilizzo di Reti neurali

2019 LINUX DAY ITALIA

FREE CIRCLE

Città di Palermo

Riconoscimento delle immagini con R e l'utilizzo di reti neurali.

Un esempio pratico.

Ing. Francesco Alaimo

Di cosa parleremo?

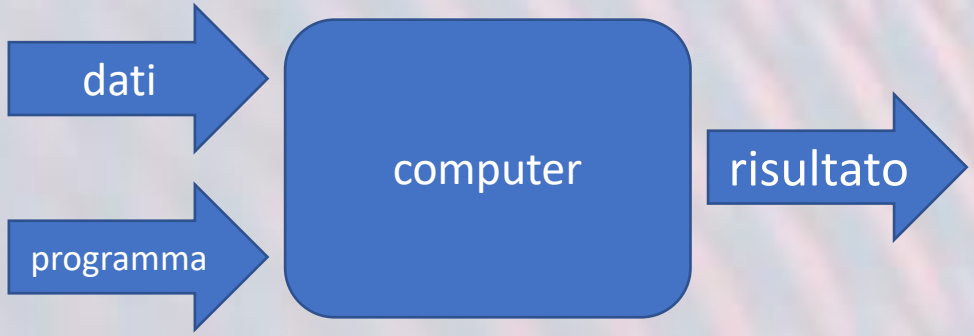
1. breve introduzione al Machine learning
2. esempio pratico di applicazione delle tecniche di machine learning utilizzando le reti neurali

Machine learning

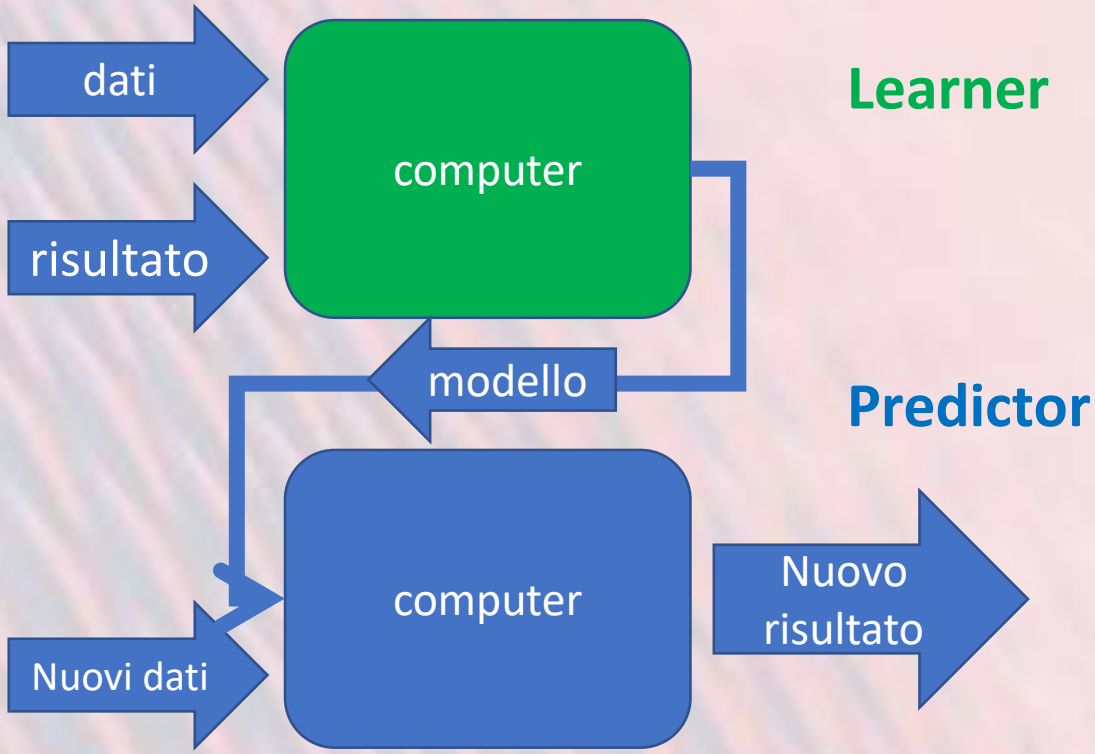
Le tecniche

Self(ie) programming ☺

Programmazione formale



Machine learning



Come impara un elaboratore? (1/4)

1. Acquisendo un set di dati (*data point*)
2. Utilizzando una strategia idonea, in funzione dell' obiettivo (*target o label*) che si vuole ottenere
3. Costruendo un *modello* che permetta di riprodurre il fenomeno analizzato
4. *Validando* il modello sulla base degli errori commessi cercando di predire le variabili *target*

Come impara un elaboratore? (2/4)

Ci sono tre strategie di apprendimento:

1. *Apprendimento supervisionato (supervised learning)*
   - ✓ *regressione:* l'obiettivo è un numero (target)
   - ✓ *Classificazione:* l'obiettivo è una classe (label)

Come impara un elaboratore? (3/4)

2. Apprendimento non supervisionato (unsupervised learning):

- ✓ *clustering:* l'obiettivo è il raggruppamento di elementi
- ✓ *riduzione della dimensionalità:* l'obiettivo è la riduzione del numero di colonne che compone il set di dati

Come impara un elaboratore? (4/4)

3. Apprendimento con rinforzo (reinforcement learning):

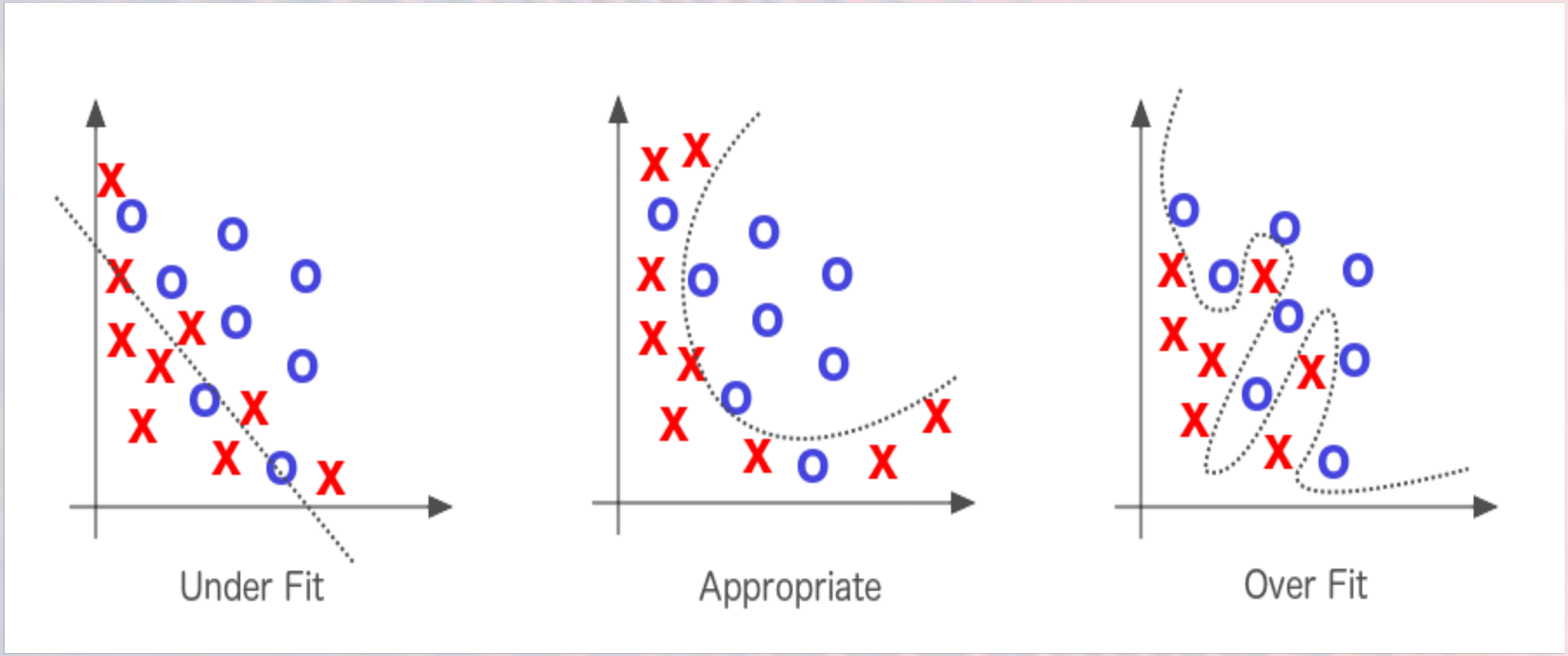
- ✓ *Sistema autonomo: l'obiettivo è un numero; rispetto alla regressione, l'algoritmo impara dagli errori e il modello migliora se si aumenta il numero di tentativi.*

Tipo di learning	Ambito di utilizzo	Esempi di algoritmo
Supervised learning	✓ Regressione lineare ✓ Classificazione	✓ Linear/logistic regression ✓ Decision tree ✓ Random forest ✓ Neural network
Unsupervised learning	✓ Clustering ✓ Riduzione della dimensionalità	✓ K-means ✓ Hierarchical clustering ✓ Latent Dirichlet Allocation ✓ Principal Component Analysis ✓ Singular Value Decomposition
Reinforcement learning	✓ Sistema autonomo	✓ Q-Lerning ✓ Monte Carlo

Underfitting e overfitting (1/2)

- Un modello può incorrere in due problematiche:
  - ✓ *Underfitting*: le predizioni sono spesso errate; il modello ha imparato da un set di dati insufficiente
  - ✓ *Overfitting*: le predizioni sono quasi sempre corrette; il modello riproduce perfettamente il fenomeno, ma generalizza male

Underfitting e overfitting (2/2)



Fonte img: Internet

Addestrare il modello: il partitioning

- Il modello migliore è detto *weel fitted*, cioè predice bene con i dati da cui ha imparato, ma anche con quelli futuri
- Per addestrare il modello si suddivide il data set in due porzioni (*partitioning*):
  - ✓ *training set:* usato per l'apprendimento
  - ✓ *testing set:* usato per la validazione

Valutazione del modello (1/3)

- Ottenuto il modello è necessario valutarne l'efficacia; ci sono diverse tecniche:
  - ✓ *RMSE Root Mean Squared Error :* adatto alla regressione

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(r_i - p_i)^2}{N}}$$

Valutazione del modello (2/3)

- ✓ *coefficiente di determinazione o* R^2*:* adatto alla regressione, permette di confrontare modelli diversi

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(r_i - p_i)^2}{\sum_{i=1}^{N}(r_i - \bar{r})^2}$$

Se:

- $R^2 \cong 0$ il modello è in *underfitting*
- $R^2 \cong 1$ il modello è in *overfitting*

Valutazione del modello (3/3)

- ✓ *Confusion matrix:* adatto alla classificazione;

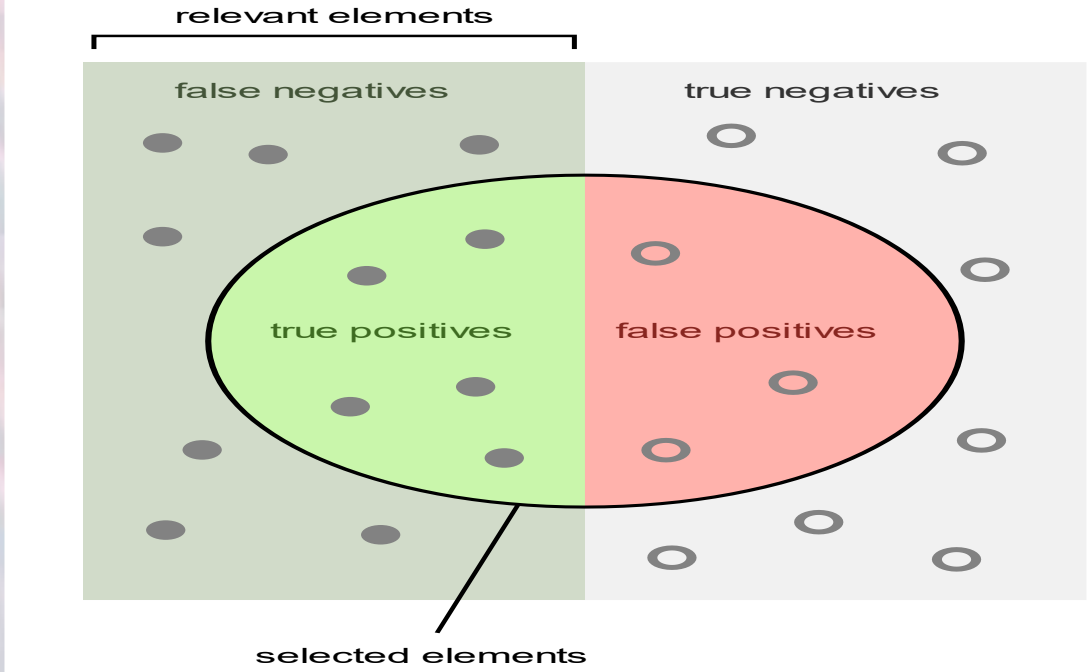
  ponendo: **TP** = True Positive, **TN** = True Negative, **FP** = False Positive, **FN** = False Negative

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN}$$

Confusion matrix, Precision e Recall

		Predicted	
		YES	NO
Real	YES	TP	FN
	NO	FP	TN



https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg

Classificazione: metodi supervisionati

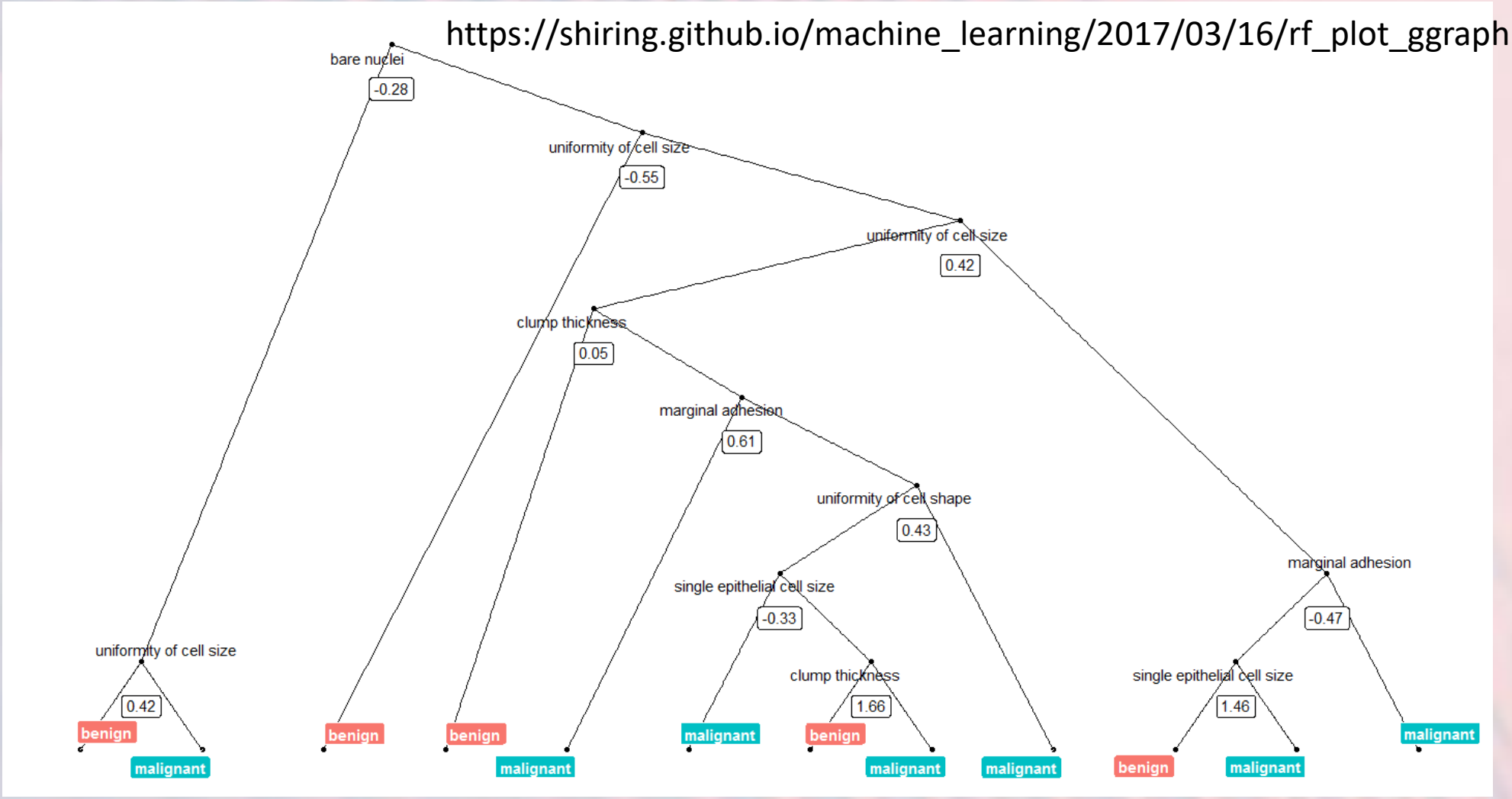
- L'algoritmo apprende da una serie di data points già associati ad un obiettivo (target o label)
  - ✓L'*addestramento* viene eseguito su una porzione del data set corredata di obiettivo
  - ✓La *predizione* viene eseguita su una porzione del data set non corredata di obiettivo (che è comunque noto)
  - ✓La *valutazione* del modello viene eseguita con uno dei metodi prima mostrati

Metodi supervisionati: decisione tree

- Si basa su processi decisionali
- genera un grafo ad albero, composto da nodi, archi e foglie, la cui esplorazione avviene tramite una sequenza domanda/risposta sui dati
- Più algoritmi decision tree possono essere usati assieme (*ensemble*) per migliorare la bontà del modello (es. nel *random forest*)

Decision tree: ovvero, l'albero delle decisioni

https://shiring.github.io/machine_learning/2017/03/16/rf_plot_ggraph



Classificazione: metodi non supervisionati

- L'algoritmo apprende da data points non associati ad un obiettivo; in pratica, deve riconoscere se vi sono delle:
  - ✓ *similarità*: per poter raggruppare gli elementi (clustering)
  - ✓ *Informazioni ridondate*: per poter ridurre il numero di colonne (*dimensionality reduction*)

metodi non supervisionati: k-means

- Si basa sulla ‘distanza’ tra gli elementi che compongono i data points
- I data point vengono associati ai gruppi (cluster) con cui hanno una distanza più bassa
- È un processo iterativo che ha come iper-parametri:
  - ✓il numero di iterazioni prima dell’arresto
  - ✓Il numero dei centroidi (baricentri dei cluster)
- E’ efficiente anche con più variabili, in spazi multi variati

k-means: funzionamento

1. Inizializzazione: vengono scelti i centroidi (in genere sono dei punti del dataset)
2. Attribuzione ai gruppi: viene calcolata la distanza di ogni punto rispetto ai centroidi e associato al centroide con minore distanza: ogni centroide definisce un cluster
3. Calcolo dei nuovi centroidi: si calcola la media delle distanze di tutti i punti associati al cluster (centroide) e ridefinito un nuovo centroide (altro punto del dataset) le cui coordinate sono baricentriche rispetto a tutte quelle dei punti del cluster
4. Si ripetono i passi 1. e 2. fino a che non si trovano nuovi centroidi 'migliori' di quelli già trovati, o non si è raggiunto il numero massimo di iterazioni

k-means: funzionamento

https://www.youtube.com/watch?v=5I3Ei69I40s

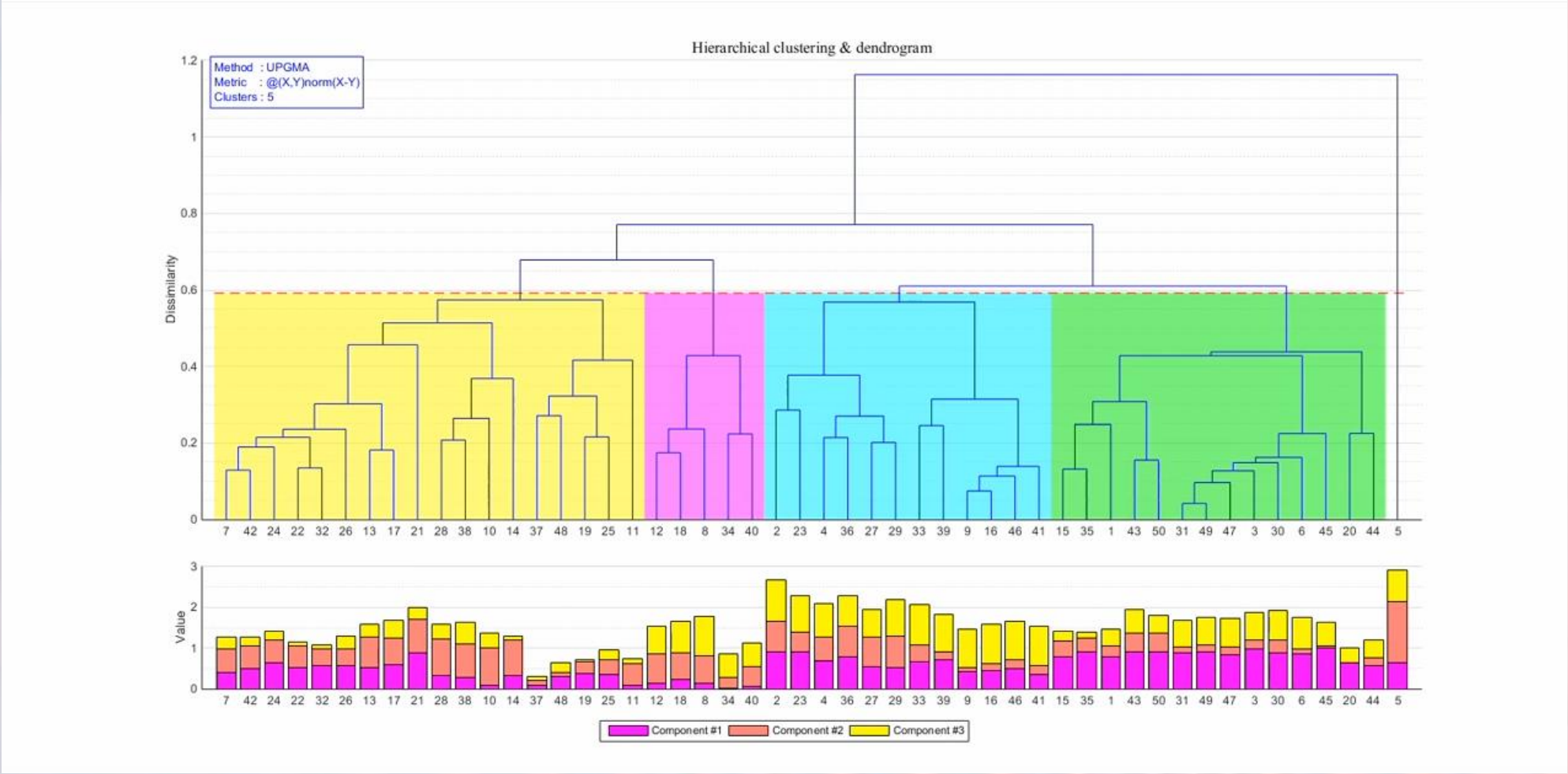
Metodi non supervisionati: Hierarchical Clustering

- Differisce dal K-means, perché trova automaticamente il numero dei cluster, tramite due diverse tecniche:
  - ✓*Agglomerative*: l'algoritmo si avvia formando tanti cluster quanti sono gli elementi, in seguito prova a ridurre il numero di cluster
  - ✓*Divisive*: l'algoritmo si avvia con un unico cluster che raggruppa tutti gli elementi, in seguito prova ad aumentare il numero di cluster
- La strategia più spesso adoperata è la prima, chiamata *HCA* (Hierarchical Agglomerative Clustering)

Hierarchical Agglomerative clustering: funzionamento

1. Inizializzazione: si calcola la matrice delle distanze di ogni punto rispetto agli altri
2. Creazione cluster: si scorre la matrice delle distanze individuando la coppia di punti (o di cluster) con la minore distanza, unendoli per formare un nuovo cluster
3. Aggiornamento della matrice delle distanze: si scorre la matrice sostituendo i cluster trovati al punto 2. ai rispettivi elementi e ricalcolando le distanze tra i punti e i cluster ottenuti. I cluster trovati costituiscono un livello di gerarchia
4. Si ripetono i punti 1. e 2. fino a quando tutti i punti sono stati associati ad un cluster

HCA



https://www.youtube.com/watch?v=EgZ2XF9rNXM

Normalizzazione

- Per poter usare k-means o HCA è necessario *normalizzare* i valori numerici
- Il metodo più comune per la normalizzazione è il *min-max*, che 'scala' i valori di ogni colonna all'interno del range 0-1 attraverso la formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- Il processo inverso è detto *denormalizzazione*

Machine learning

Alcuni Tools

R (1/2)

- È un ambiente ed un linguaggio di sviluppo per l'analisi statistica, tra i più utilizzati al mondo, che deriva da S (un programma commerciale)
- È rilasciato con licenza GNU GPL e disponibile per i principali SO
- È supportato da una community molto attiva e può essere potenziato grazie a moduli aggiuntivi (i packages)
- Tramite i packages R può supportare hadoop e altre piattaforme di BIG DATA e data stream, facebook, twitter, text mining, GIS e molto, molto altro

R (2/2)

RGui (64-bit)

File Modifica Pacchetti Finestre Aiuto

R Console

```
+         bodies_text<-paste(bodies_text,articolo.body
+       }
+       ## accodo l'articolo a quelli trovati finora
+       if(!is.null(bodies_text)){
+         result_articoli<-append(result_articoli,bodi
+         ## variabile di controllo che serve a memori
+         result_ids<-append(result_ids,r_id)
+         ## variabile di controllo che visualizza il
+         cdocs<-cdocs+1
+         ## print per visualizzare avanzamento fetchi
+         print(c(r_id,cdocs))
+         bodies_text<-NULL
+       }else{
+         print("Body vuoto, scartato!")
+       }
+
+     }else{
+       print(paste("Scartato documento con id",r_id,s
+     }
+   }
+   print(paste("Elaborati",cdocs,"scartati",(count-cd
+   return(result_articoli)
+ }
>
>
```

C:\Users\falai\OneDrive\Documenti\tesi.R - Editor di R

```
## installazione e caricamento delle librerie
require("topicmodels")
require("rentrez")
require("tm")
require("XML")
require("wordcloud2")
require("slam")
require("tau")

library(topicmodels)
library(rentrez)
library(tm)
library(XML)
library(wordcloud2)
library(slam)
library(tau)

##definizione del database da utilizzare
database<-"pmc"
## definizione delle proprie stopwords
my_stop_w<-c("the","this","thus","figure","patient","patients","study","studies
## definizione della query di ricerca
query<-c("(Alzheimer's disease[ARTI] AND Alzheimer's disease[MAJR] ) AND \"oper
## determinazione del numero massimo di documenti presenti sul database in base
count=entrez_search(db=database,term=query,retmax=0)$count
```

Rstudio (1/3)

- È l'IDE per eccellenza di R, disponibile con licenza commerciale e open source
- Esiste una versione desktop per i principali SO e una server solo per Linux
- Arricchisce l'ambiente nativo di R di numerose caratteristiche, come:
  - ✓Evidenziazione della sintassi, completamento del codice e indentazione
  - ✓Esecuzione di R direttamente dall'IDE
  - ✓Navigazione tra le funzioni, help integrato, gestione di più progetti

Rstudio (2/3)

- ✓Visualizzazione dei dati
- ✓Debug interattivo
- ✓Supporto del versioning con GIT e Subversion
- ✓Produzione di documenti in HTML, PDF, Word e slide show
- ✓Supporto di grafici interattivi con i packages Shiny e ggvis

Rstudio (3/3)

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

NeuralNetwork.R | convolutional_nn_tutorial_2(64).R | convolutional_nn_tutorial_3(64).R

```
51 # Set s[illegible]
52 mx.set.[illegible](1)
53
54 # Device used. CPU in my case.
55 devices <- mx.cpu()
56
57 # Training
58 #------------------------------------------------------------------------------
59
60 # Train the model
61 model <- mx.model.FeedForward.create(NN_model,
62                                      X = train_array,
63                                      y = train_y,
64                                      ctx = devices,
65                                      num.round = 240,
66                                      array.batch.size = 40,
67                                      learning.rate = 0.01,
68                                      momentum = 0.9,
69                                      eval.metric = mx.metric.accuracy,
70                                      epoch.end.callback = mx.callback.log.train.metric(100))
71
72 # Testing
73 #------------------------------------------------------------------------------
74
75 # Predict labels
76 predicted <- predict(model, test_array)
77 # Assign labels
78 predicted_labels <- max.col(t(predicted)) - 1
79 # Get accuracy
80 sum(diag(table(test[, 1], predicted_labels)))/40
81
82 ##############################################################################
```

78:21 (Untitled) R Script

Console | Terminal | Jobs

C:/Users/falai/

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from C:/Users/falai/.RData]

Loading required package: EBImage
>
```

Environment | History | Connections

Import Dataset | Grid | Global Environment

Name	Type	Length	Size	Value
data	data.frame	6	3.7 KB	75 obs. of 6 variables
datatest	data.frame	6	1.8 KB	10 obs. of 6 variables
datatrain	data.frame	6	2.9 KB	45 obs. of 6 variables
i	integer	1	56 B	100L
img	Image	0	0 B	Formal class Image
img_matrix	matrix	0	0 B	num [1:64, 1:64] 0.517 0.463 0.281 …
img_vector	numeric	4096	32 KB	num [1:4096] 0.517 0.517 0.517 0.496 0…
index	integer	65	312 B	int [1:65] 28 18 46 64 73 72 71 15 23 …
j	integer	1	56 B	65L
k	numeric	1	56 B	100
label	integer	1	56 B	39L
labels	data.frame	1	2.3 KB	400 obs. of 1 variable
List	list	65	46.9 KB	List of 65
Matrix.RMSE	matrix	5600	44 KB	num [1:100, 1:56] 9.62 8.98 6.66 8.…
max	numeric	6	648 B	Named num [1:6] 160 6 5 320 14 ...

Files | Plots | Packages | Help | Viewer

Zoom | Export

Python (1/2)

- È un linguaggio free, interpretabile, con una comunità molto attiva, disponibile per diverse piattaforme
- Supporta la programmazione procedurale e ad oggetti
- Dispone di ricca dotazione di librerie base e molte altre, per gli usi più svariati
- I programmi vengono compilati in bytecode prima dell'esecuzione (come in Java), quindi è abbastanza performante

Python (2/2)

- Utilizza il garbage collection per gestire la memoria (come in Java)
- Può integrarsi con altri linguaggi, come il framework .NET (IronPython) e Java (Jython)
- È un linguaggio versatile che si presta al machine learning grazie alla presenza di numerose librerie come TensorFlow, scikit-learn, NumPy, SciPy

R Vs Python

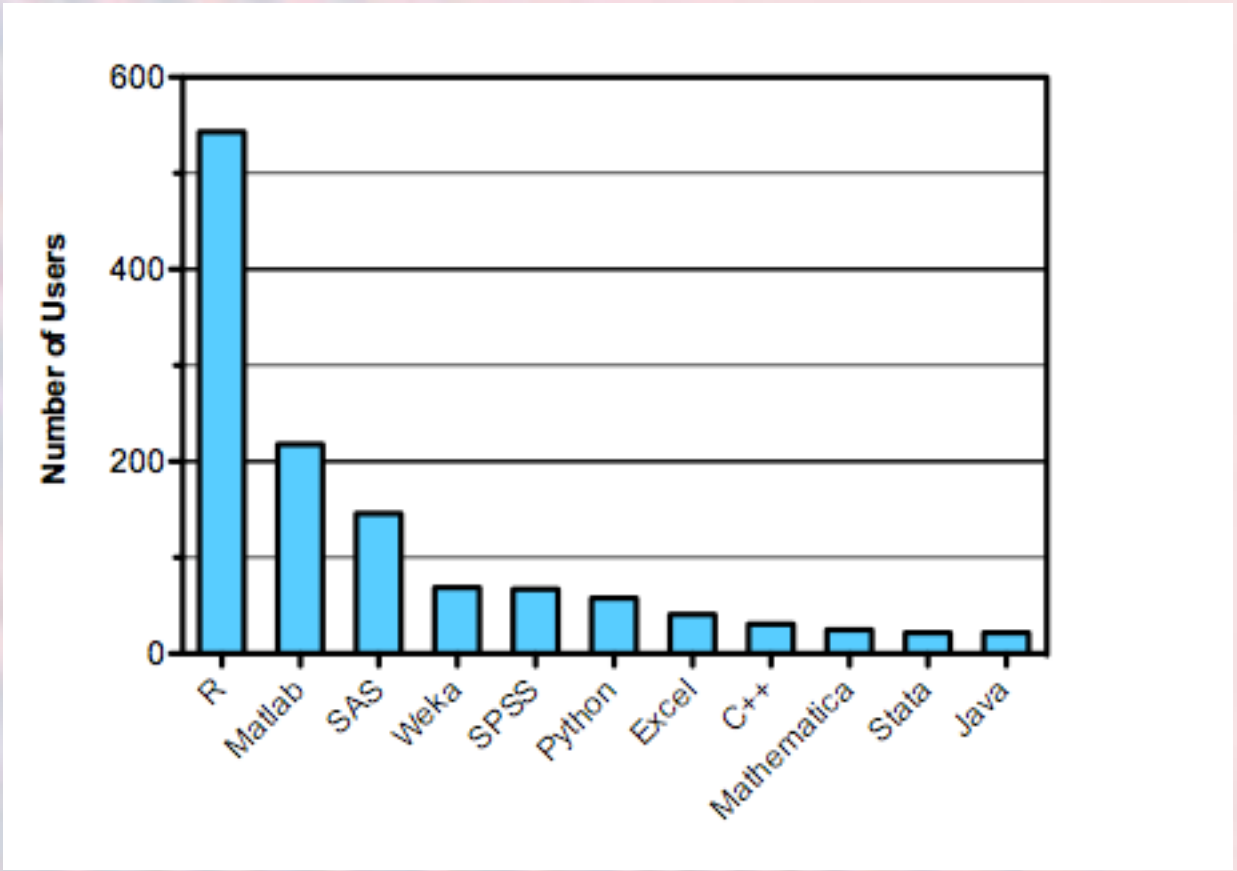
https://www.differencebetween.com/difference-between-r-and-vs-python/

R	Python
R is a programming language and software environment for statistical computing, graphics representation and reporting.	Python is an interpreted high-level programming language for general purpose programming.
Developed By	
R is supported by the R Foundation for Statistical Computing.	Python is supported by the Python Software Foundation.
Data Structures	
R supports data structures such as vectors, lists, matrices, arrays, factors and data frames.	Python supports data structure such as lists, dictionaries and tuples.
Switch Statement	
R supports switch statement.	Python does not support switch statement.
Scripts	
R scripts end with. R extension.	Python scripts end with .py extension.
IDE	
The common IDE for R programming is RStudio.	The common IDEs for Python programming are PyCharm and Eclipse.
Applications	
R can be used for statistical computing, machine learning and data analytics.	Python can be used for multiple applications such as machine learning, web development, networking, scientific computing, automation, natural language processing, etc.

R Vs Python: most used (secondo Kaggle)

What programming/statistics languages you used for an analytics / data mining / data science work in 2013? [713 votes total]

	% users in 2013	% users in 2012
R (434 voters in 2013)	60.9%	52.5%
Python (277)	38.8%	36.1%
SQL (261)	36.6%	32.1%
SAS (148)	20.8%	19.7%
Java (118)	16.5%	21.2%
MATLAB (89)	12.5%	13.1%



https://machinelearningmastery.com/best-programming-language-for-machine-learning/

Esempio pratico di machine learning con R (e Python)

(sporchiamoci le mani…)

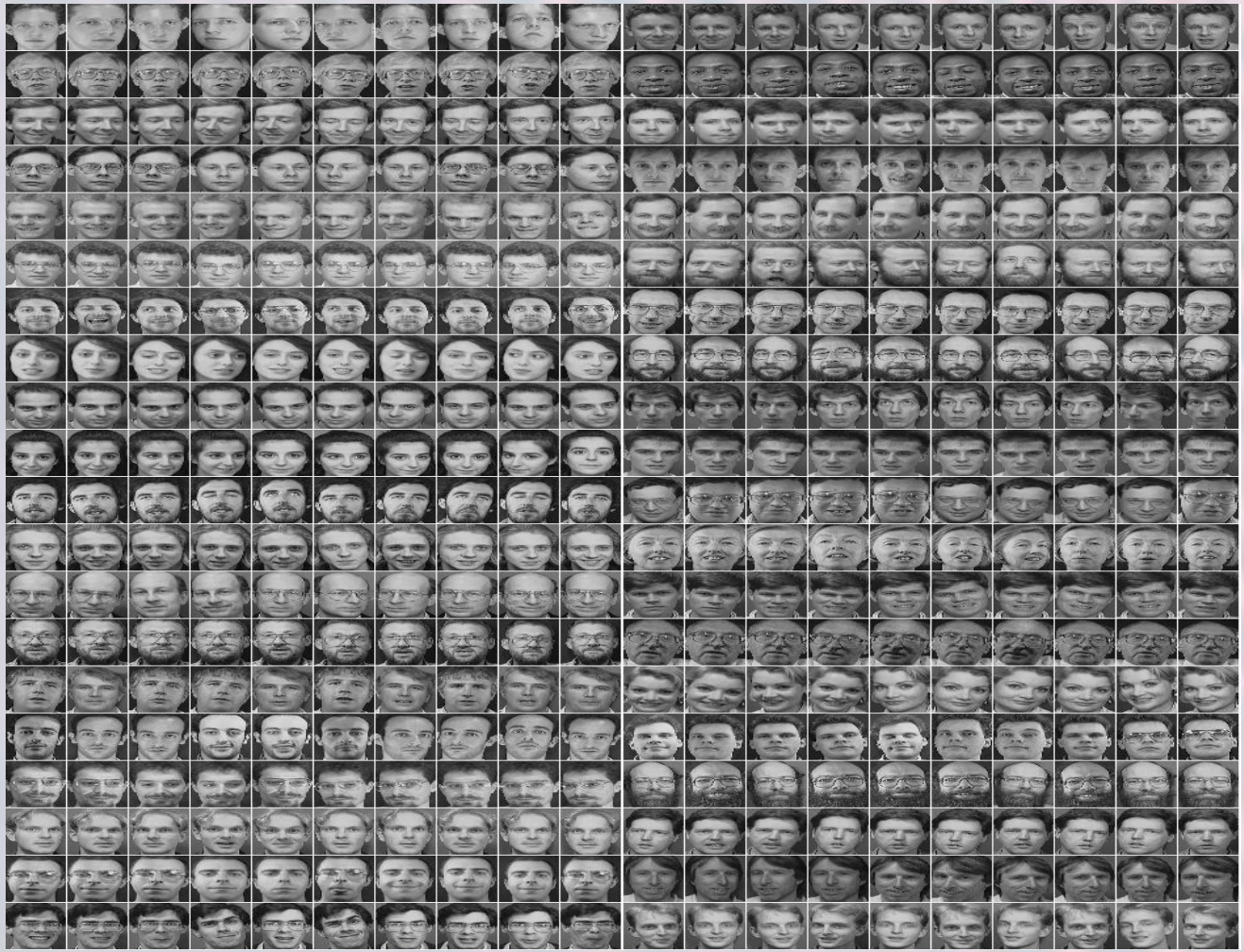
Bello tutto…Ma che ci faccio?

- Prendendo spunto da un articolo presente sul sito *https://www.r-bloggers.com*, mostreremo un esempio di utilizzo di machine learning applicato al riconoscimento delle immagini
- L'esempio è reperibile al link:
  - ✓https://www.r-bloggers.com/image-recognition-tutorial-in-r-using-deep-convolutional-neural-networks-mxnet-package/

Esempio pratico di machine learning con R (e Python)

Fonte dati

Olivetti faces



Collezione di 400 foto, creata da AT&T Laboratories Cambridge, per un progetto di riconoscimento facciale

https://www.cl.cam.ac.uk/research/dtg/attarchive/facesataglance.html

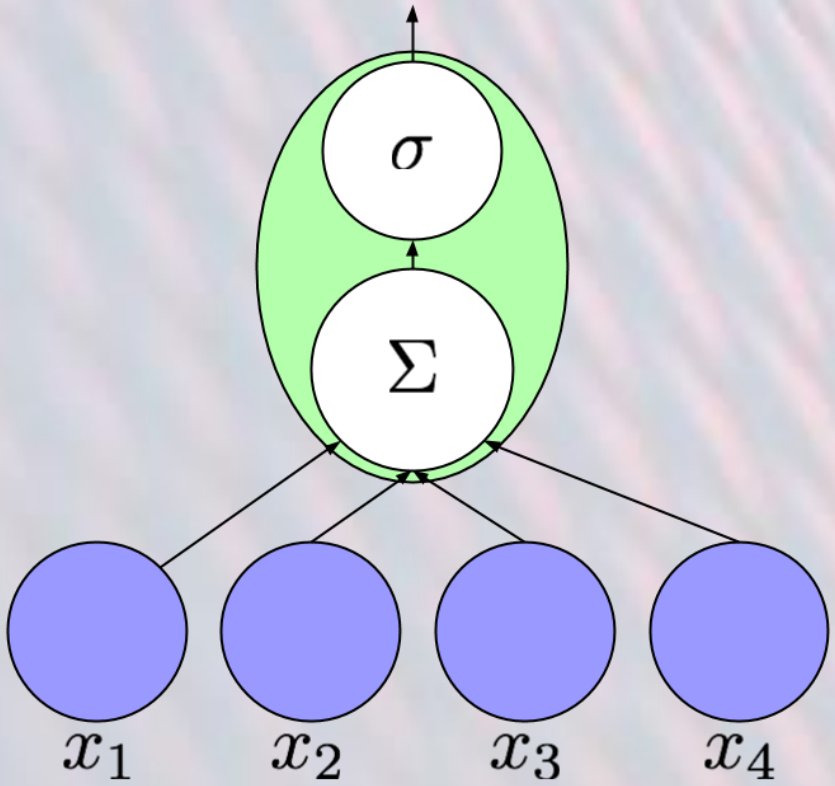
Esempio pratico di machine learning con R (e Python)

Concetti, Tools, librerie

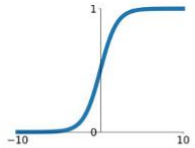
MXNet

- *MXNet* è una framework per il *deep learning* basato sulle reti neurali
- Il *deep learning* è alla base della guida autonoma, del riconoscimento vocale, della traduzione automatica e altro
- Le reti neurali cercano di riprodurre, in hardware e in software, i processi che avvengono nel cervello umano

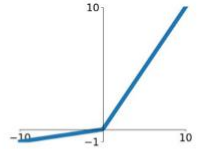
Il Neurone formale di Mc Culloch and Pitts



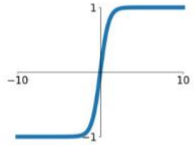
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



Leaky ReLU
$\max(0.1x, x)$

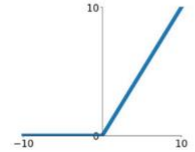


tanh
$\tanh(x)$

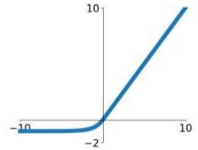


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ReLU
$\max(0, x)$



ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

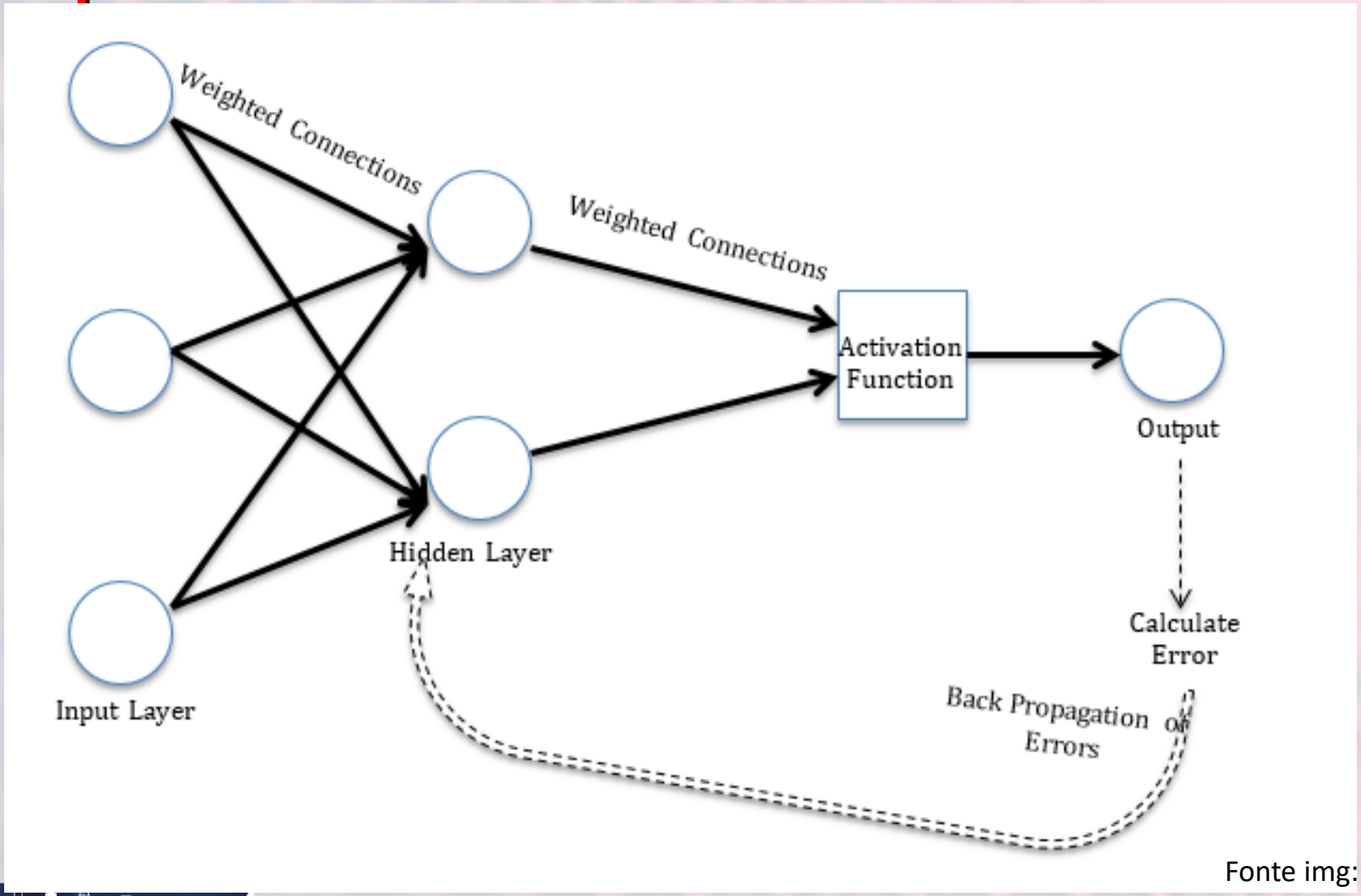


https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

Come funzionano un neurone formale e una rete neurale?

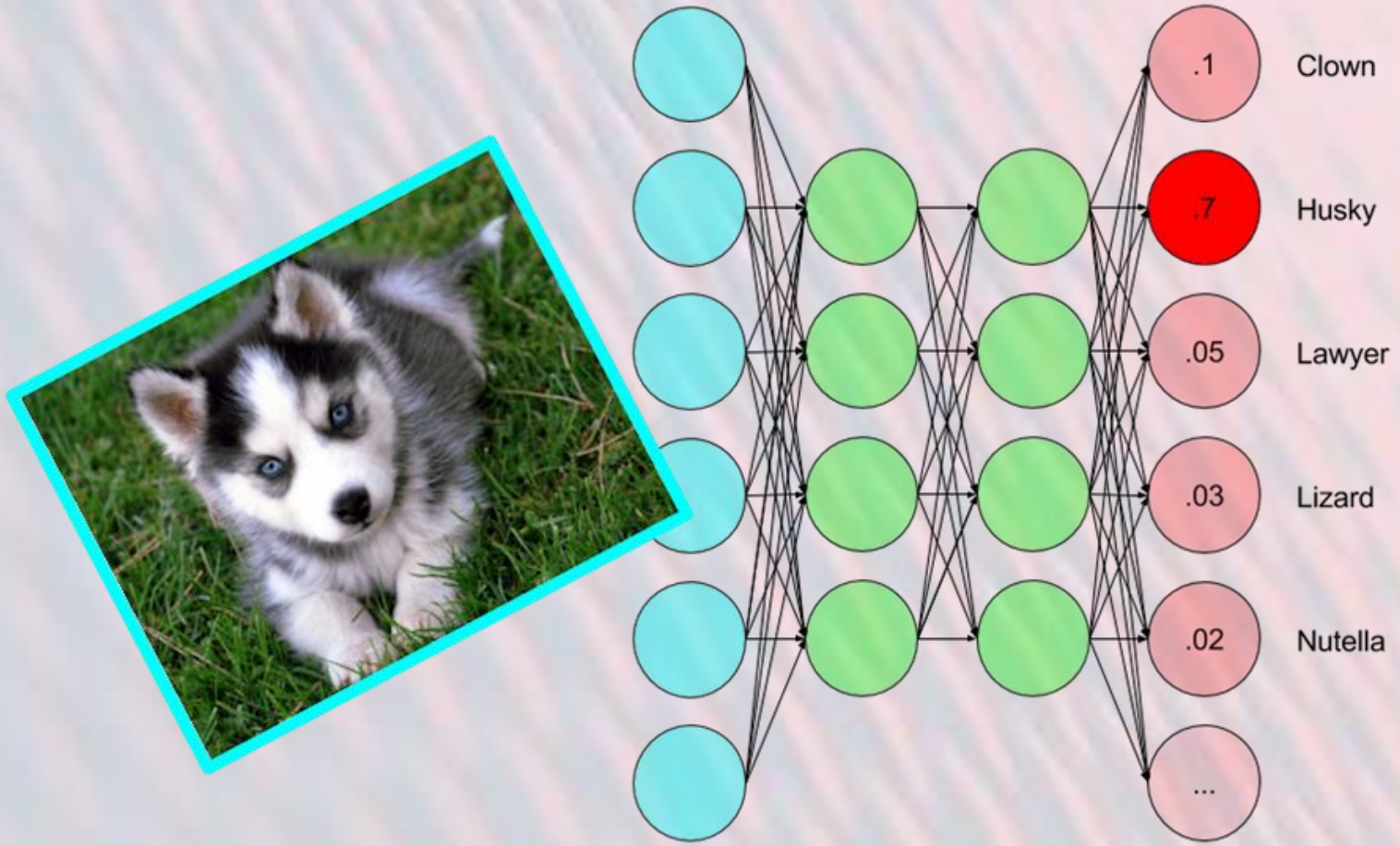
- Quando la combinazione lineare degli ingressi (x) supera una soglia di attivazione, in funzione di un peso e della *funzione di attivazione (σ)*, il neurone si *attiva*, altrimenti è *inibito*
- Le caratteristiche di una rete neurale dipendono dal numero di layer, dalle connessioni e dalla funzione di attivazione
- La rete *apprende* calcolando il valore dei pesi che determinano l'uscita col minore errore
- Alcune reti neurali sono dotate di un ingresso retro azionato (*back propagation*), che ne migliora le prestazioni

Un semplice modello di rete neurale



Fonte img: Internet

Che razza di cane??



Fonte img: http://mxnet.incubator.apache.org/versions/master/faq/why_mxnet.html

Come avviene il riconoscimento delle immagini?

- L'immagine viene trasformata in una matrice che, per ogni pixel, memorizza informazioni sulla luminosità e il colore
- Ogni matrice viene associata ad una *classe* e passata in input ad una rete neurale per la fase di apprendimento (*training*)
- Al termine di questa fase, nuove matrici vengono passate al modello ottenuto per valutarne l'efficacia (*testing*)
- Se è presente una linea di *back propagation*, i pesi vengono aggiustati automaticamente per ridurre l'errore in uscita
- Quando l'errore è accettabile, il modello sarà pronto

EBImage

- È un pacchetto per R, distribuito come parte del progetto *Bioconductor*, per la lettura, la scrittura e la visualizzazione delle immagini
- Consente di trattare le immagini come un oggetto di tipo array, comprensivo di informazioni sull'intensità dei pixel, per potervi operare con applicazioni di algebra lineare
- Supporta immagini in scala di grigio o a colori

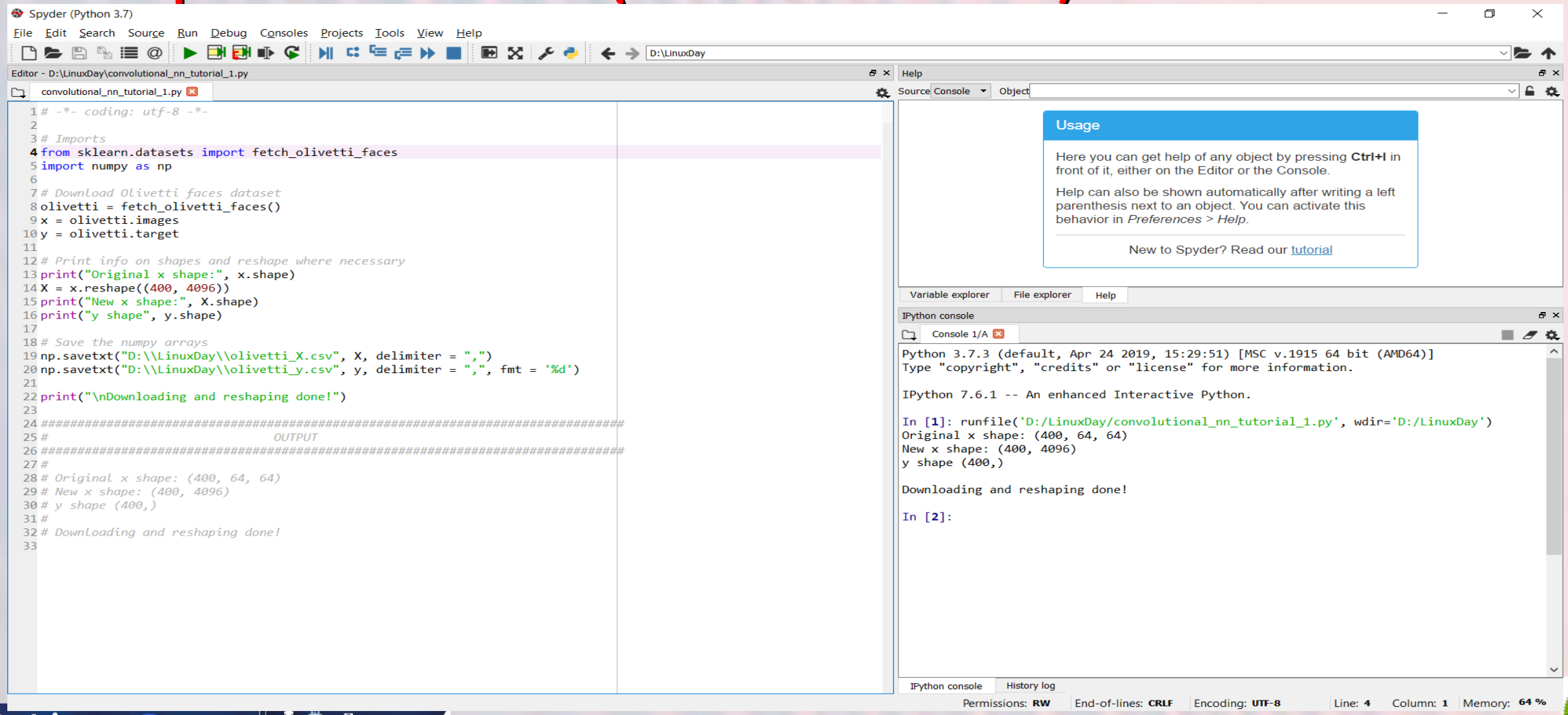
Esempio pratico di machine learning con R (e Python)

Codice

Acquisizione dati (con python)

```
from sklearn.datasets import fetch_olivetti_faces # Imports
import numpy as np
olivetti = fetch_olivetti_faces() # Download Olivetti faces dataset
x = olivetti.images
y = olivetti.target
print("Original x shape:", x.shape) # Print info on shapes and reshape where necessary
X = x.reshape((400, 4096))
print("New x shape:", X.shape)
print("y shape", y.shape)
np.savetxt("D://LinuxDay//olivetti_X.csv", X, delimiter = ",") # Save the numpy arrays
np.savetxt("D://LinuxDay //olivetti_y.csv", y, delimiter = ",", fmt = '%d')
print("\nDownloading and reshaping done!")
```

Acquisizione dati (con anaconda)



Preparazione dati in R (1/3)

```
rm(list=ls())

require(EBImage) # Load EBImage library

X <- read.csv("d:\\linuxDay\\olivetti_X.csv",header = F)
# Load data

labels <- read.csv("d:\\linuxDay\\olivetti_y.csv",header
= F)

rs_df <- data.frame() # Dataframe of images

# Main loop: for each image, set it to greyscale

for(i in 1:nrow(X))
```

Preparazione dati in R (2/3)

```
{result <- tryCatch({# Try-catch
img <- as.numeric(X[i,]) # Image (as 1d vector)
img <- Image(img, dim=c(64, 64), colormode = "Grayscale") # Reshape as a 64x64 image (EBImage object)
    img_matrix <- img@.Data
img_vector <- as.vector(t(img_matrix)) # Coerce to a vector
label <- labels[i,] # Add label
   vec <- c(label, img_vector)
rs_df <- rbind(rs_df, vec) # Stack in rs_df using rbind
print(paste("Done",i,sep = " "))}, # Print status
error = function(e){print(e)}) # Error function (just prints the error). }
```

Preparazione dati in R (3/3)

```
# Set names. The first columns are the labels, the other columns are the pixels.
names(rs_df) <- c("label", paste("pixel", c(1:4096))).
set.seed(100) # Set seed for reproducibility purposes
shuffled <- rs_df[sample(1:400),] # Shuffled df
train_64 <- shuffled[1:360, ] # Train-test split
test_64 <- shuffled[361:400, ]
# Save train-test datasets
write.csv(train_64, "d:\\linuxDay\\train_64.csv", row.names = FALSE)
write.csv(test_64, "d:\\linuxDay\\test_64.csv", row.names = FALSE)
```

Preparazione dei dati

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

convolutional_nn_tutorial_2(64).R | convolutional_nn_tutorial_3(64).R

```
15 for(i in 1:nrow(X))
16 {
17   # Try-catch
18   result <- tryCatch({
19     # Image (as 1d vector)
20     img <- as.numeric(X[i,])
21     #plot vector face
22     #plot(img)
23     # Reshape as a 64x64 image (EBImage object)
24     img <- Image(img, dim=c(64, 64), colormode = "Grayscale")
25     #plot current face
26     plot(img)
27     Sys.sleep(1)
28     img_matrix <- img@.Data
29     # Coerce to a vector
30     img_vector <- as.vector(t(img_matrix))
31     # Add label
32     label <- labels[i,]
33     vec <- c(label, img_vector)
34     # Stack in rs_df using rbind
35     rs_df <- rbind(rs_df, vec)
36     # Print status
37     print(paste("Done",i,sep = " "))},
38     # Error function (just prints the error). Btw you should get no errors!
39     error = function(e){print(e)})
40 }
41
42
43 # Set names. The first columns are the labels, the other columns are the pixels.
44 names(rs_df) <- c("label", paste("pixel", c(1:4096)))
45
46 # Train-test split
```

Console | Terminal | Jobs

```
[1] "Done 158"
[1] "Done 159"
[1] "Done 160"
[1] "Done 161"
[1] "Done 162"
[1] "Done 163"
[1] "Done 164"
[1] "Done 165"
[1] "Done 166"
[1] "Done 167"
[1] "Done 168"
[1] "Done 169"
[1] "Done 170"
[1] "Done 171"
[1] "Done 172"
[1] "Done 173"
[1] "Done 174"
[1] "Done 175"
[1] "Done 176"
[1] "Done 177"
[1] "Done 178"
```

Environment | History | Connections

Name	Type	Length	Size	Value
i	integer	1	56 B	12L
img	Image	0	0 B	Formal class Image
img_matrix	matrix	0	0 B	num [1:64, 1:64] 0.541 0.587 0.64 0...
img_vector	numeric	4096	32 KB	num [1:4096] 0.541 0.554 0.579 0.583 0...
label	integer	1	56 B	1L
labels	data.frame	1	2.3 KB	400 obs. of 1 variable
result	character	1	112 B	"Done 11"
rs_df	data.frame	4097	1.1 MB	11 obs. of 4097 variables
vec	numeric	4097	32.1 KB	num [1:4097] 1 0.541 0.554 0.579 0.583...
X	data.frame	4096	13 MB	400 obs. of 4096 variables

Files | Plots | Packages | Help | Viewer

Generazione del modello in R (1/4)

```
rm(list=ls())
require(mxnet) # Load MXNet
train <- read.csv("d:\\LinuxDay\\train_64.csv") # Loading data train and test datasets
test <- read.csv("d:\\LinuxDay\\test_64.csv")
train <- data.matrix(train) # Set up train and test datasets
train_x <- t(train[, -1])
train_y <- train[, 1]
train_array <- train_x
dim(train_array) <- c(64, 64, 1, ncol(train_x))
test_x <- t(test[, -1])
test_y <- test[, 1]
test_array <- test_x
dim(test_array) <- c(64, 64, 1, ncol(test_x))
```

Generazione del modello in R (2/4)

```
data <- mx.symbol.Variable('data') # Set up the symbolic model
conv_1 <- mx.symbol.Convolution(data = data, kernel = c(5, 5), num_filter = 20) # 1st conv. Lay.
tanh_1 <- mx.symbol.Activation(data = conv_1, act_type = "tanh")
pool_1 <- mx.symbol.Pooling(data = tanh_1, pool_type = "max", kernel = c(2, 2), stride = c(2, 2))
conv_2 <- mx.symbol.Convolution(data = pool_1, kernel = c(5, 5), num_filter = 50) # 2nd conv. Lay
tanh_2 <- mx.symbol.Activation(data = conv_2, act_type = "tanh")
pool_2 <- mx.symbol.Pooling(data=tanh_2, pool_type = "max", kernel = c(2, 2), stride = c(2, 2))
flatten <- mx.symbol.Flatten(data = pool_2) # 1st fully connected layer
fc_1 <- mx.symbol.FullyConnected(data = flatten, num_hidden = 500)
tanh_3 <- mx.symbol.Activation(data = fc_1, act_type = "tanh")
fc_2 <- mx.symbol.FullyConnected(data = tanh_3, num_hidden = 40) # 2nd fully connected layer
NN_model <- mx.symbol.SoftmaxOutput(data = fc_2) # Output. Softmax
```

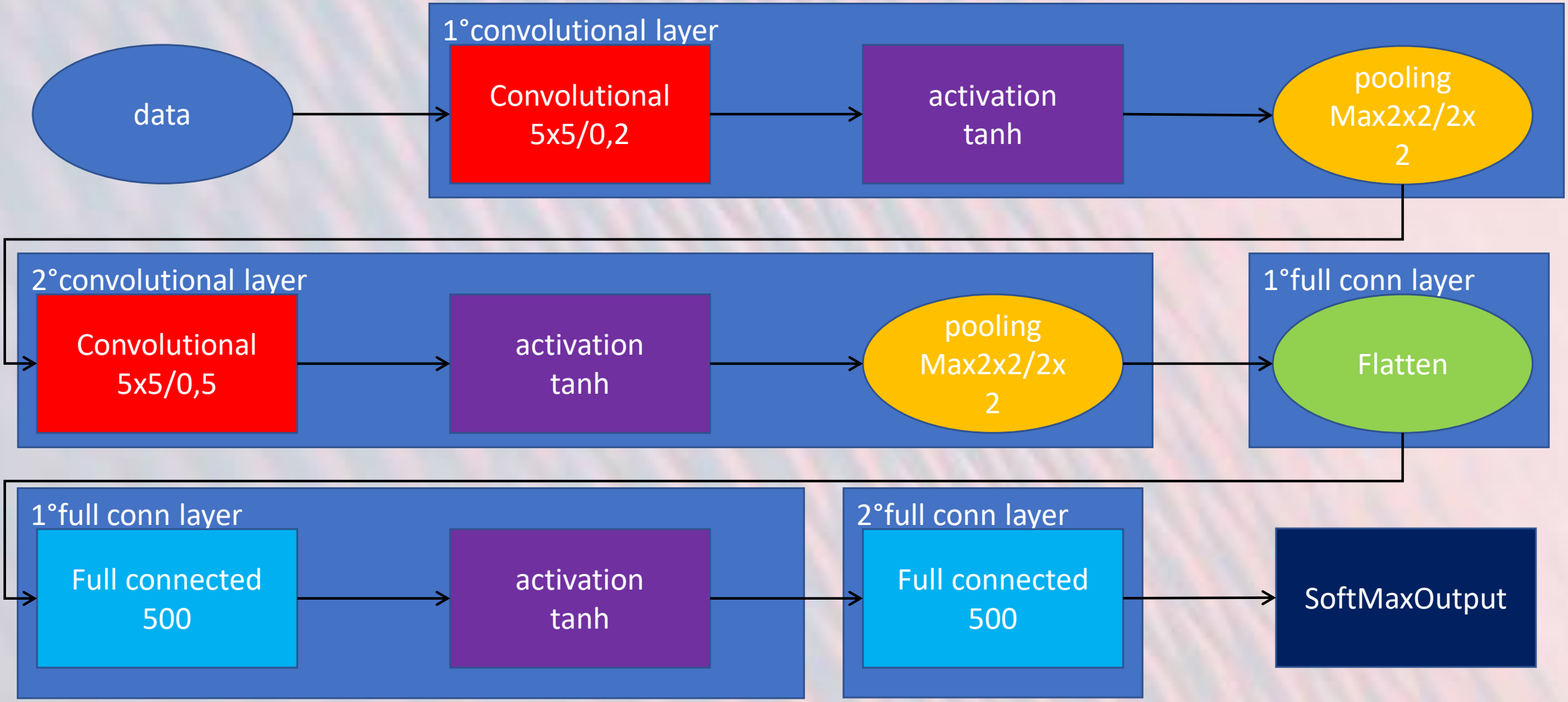
Generazione del modello in R (3/4)

```
mx.set.seed(100) # Pre-training set up , Set seed for reproducibility
devices <- mx.gpu() # Device used. GPU in my case.
model <- mx.model.FeedForward.create(NN_model, # Training , Train the model
                                     X = train_array,
                                     y = train_y,
                                     ctx = devices,
                                     num.round = 240,
                                     array.batch.size = 40,
                                     learning.rate = 0.01,
                                     momentum = 0.9,
                                     eval.metric = mx.metric.accuracy,
                                     epoch.end.callback = mx.callback.log.train.metric(100))
```

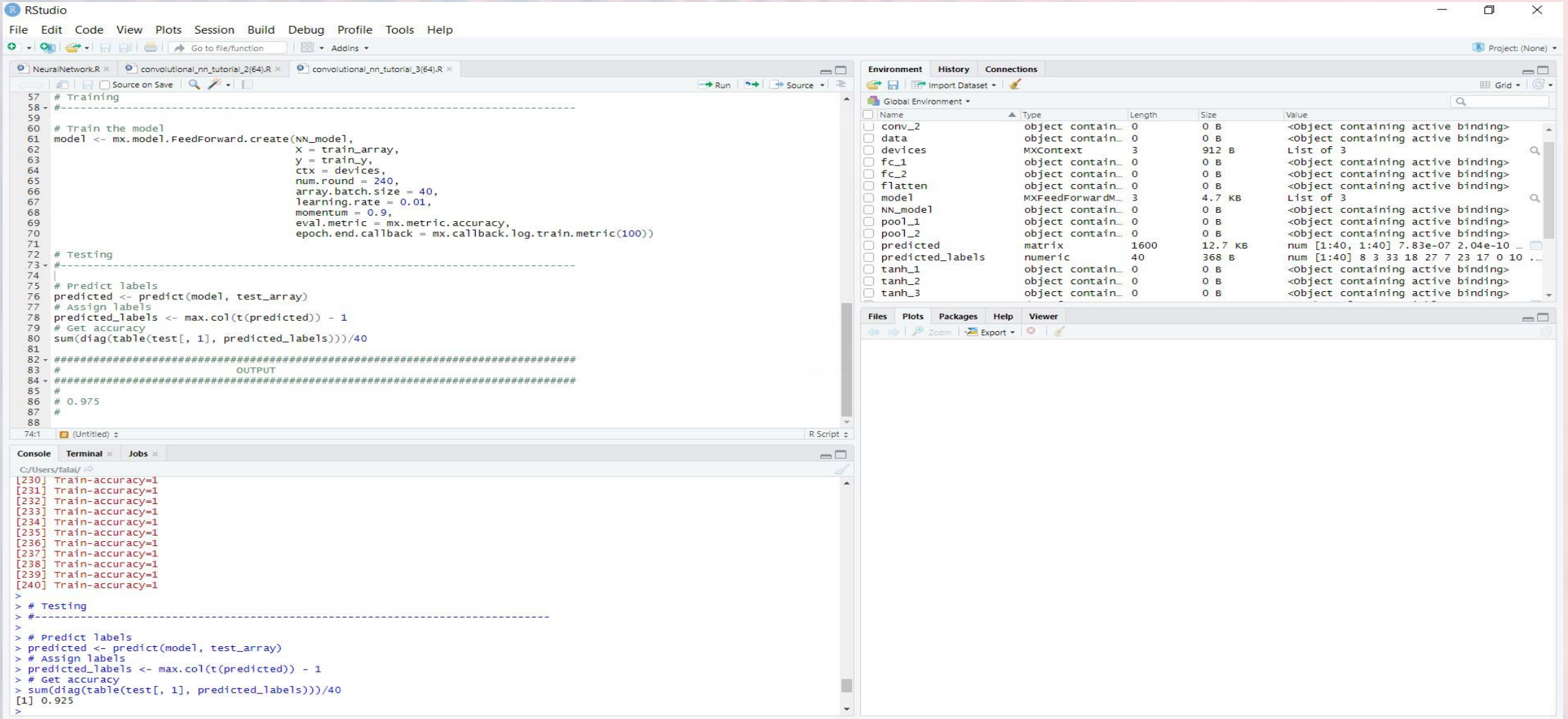
Generazione del modello in R (4/4)

```
predicted <- predict(model, test_array) # Testing, Predict labels
predicted_labels <- max.col(t(predicted)) – 1 # Assign labels
sum(diag(table(test[, 1], predicted_labels)))/40 # Get accuracy
```

Modello di rete neurale utilizzato



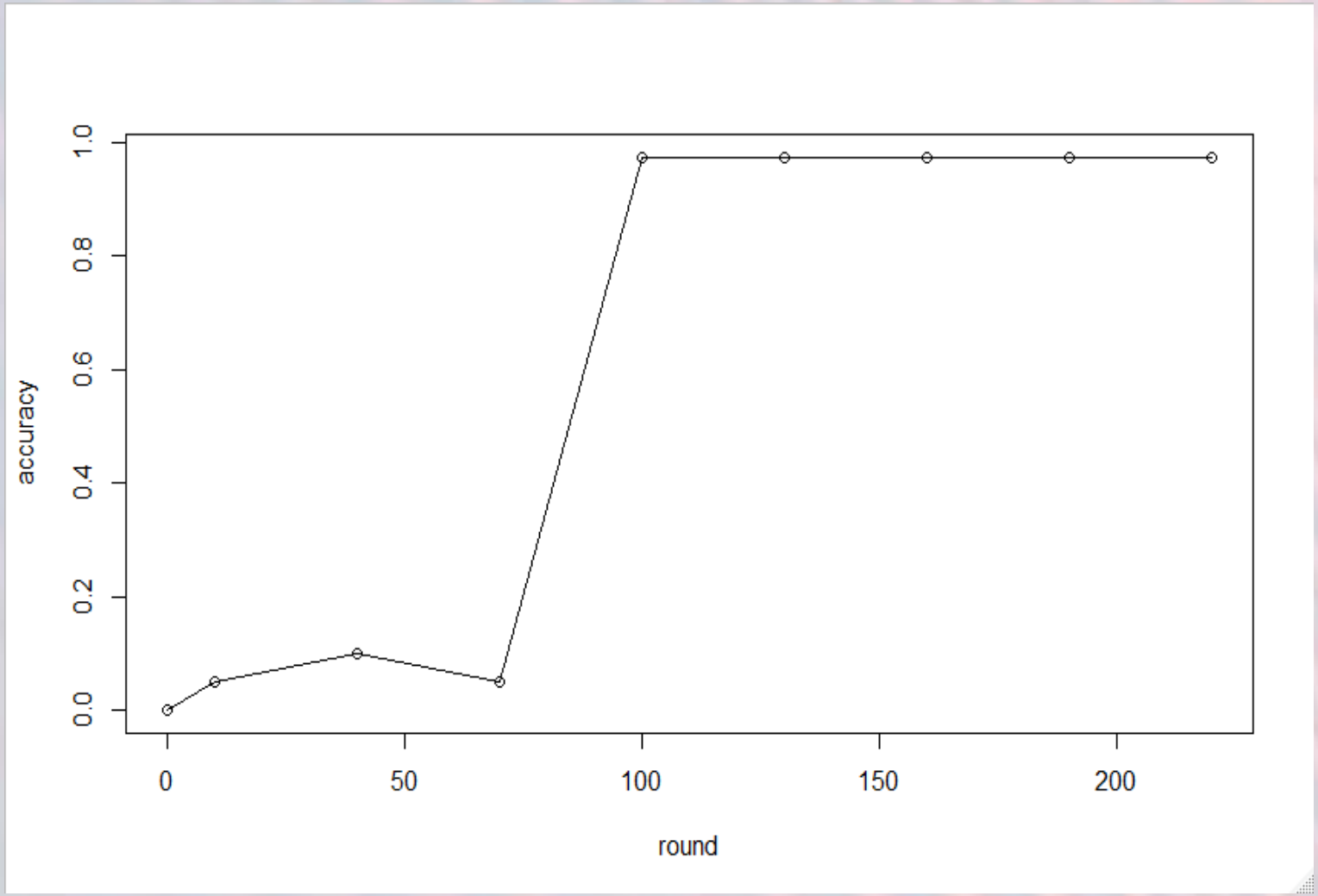
Generazione del modello



Esempio pratico di machine learning con R (e Python)

Risultati

Risultati (1/2)



Il modello di predizione raggiunge l'accuratezza del **97.5%** a circa 95 round, dimostrandosi molto accurato.

Risultati (2/2)

test labels \ predicted lables	0	1	2	5	6	7	9	11	14	16	17	19	21	24	25	27	29	30	33	34	35	36	39
0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Conclusioni

- Le reti neurali si prestano molto bene per il riconoscimento delle immagini
- Necessitano di molte risorse per funzionare (meglio usare GPU)
- Hanno lo svantaggio di adattarsi molto bene ai dati, perciò possono incorrere nel fenomeno dell'overfitting
- Di contro, necessitano di molti dati e di un numero discreto di cicli (*epoche)* di apprendimento per non incorrere nell'underfitting

Linux Day

26 Ottobre 2019

linuxday.thefreecircle.org/2019/

machinelearningwithr@gmail.com
https://t.me/MachineLearningWithR

Grazie

2019 LINUX DAY ITALIA

FREE CIRCLE

Città di Palermo