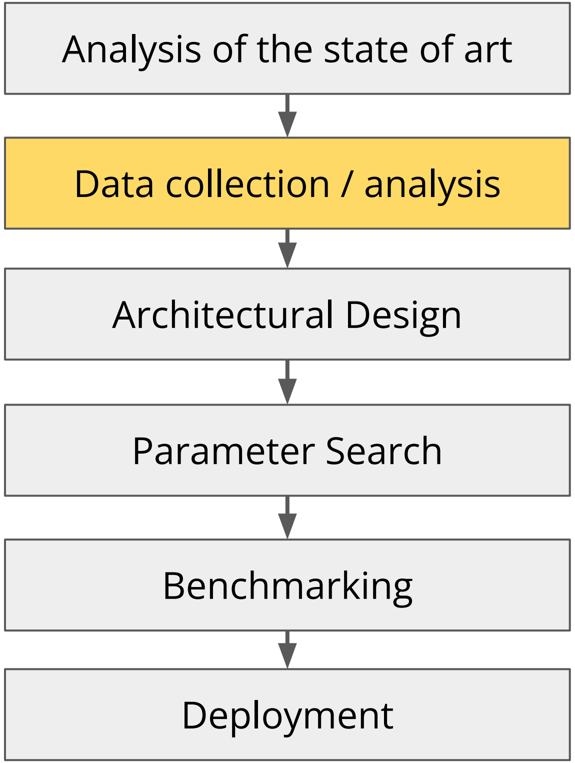
Università degli
Studi di Milano-Bicocca

Summing up

Prof. Flavio Piccoli

a.a. 2022-2023

R&D process


Analysis of the state of art

Data collection / analysis

Architectural Design

Parameter Search

Benchmarking

Deployment

pandas

scikit learn

Data preprocessing

Cleaning

Deal with missing data

↓

Encoding

Convert data in a machine-learning suitable format

↓

Normalizing

Normalize the variables

↓

Splitting

Split in train / validation / test

Cleaning

What do we do with missing data?

Three possible strategies:

- discard feature having missing data
- discard samples having missing data
- substitute missing data with plausible content
    - **booleans / categorical**: replace with mode
    - **integers**: replace with median
    - **floats**: interpolate

Encoding of input categorical variables

- Machine learning models can only work with numerical values
- It is necessary to transform the categorical values of the relevant features into numerical ones

One-hot encoding (for input variables)

brand
Fiat
BMW
Lamborghini
Fiat

→

brand_fiat	brand_bmw	brand_lambo
1	0	0
0	1	0
0	0	1
1	0	0

Encoding of target categorical variables

- Machine learning models can only work with numerical values
- It is necessary to transform the categorical values of the relevant features into numerical ones

Labeling (for estimated variables)

brand
Fiat
BMW
Lamborghini
Fiat

→

brand
0
1
2
0

Normalizing

- The range of the variables affect their importance
- We need to normalize them so that each variable resides in the same range

    - **min - max normalization**
        - if the variable under analysis has a specific range, it’s possible to use this normalization
        - `from sklearn.preprocessing import MinMaxScaler`

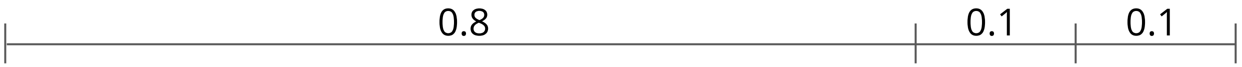
    - **standardization**
        - if the range is unknown a priori
        - sets the mean to 0 and the variance to 1
        - `sklearn.preprocessing import StandardScaler`

Splitting

It is possible to split in train, validation and test with the following code:

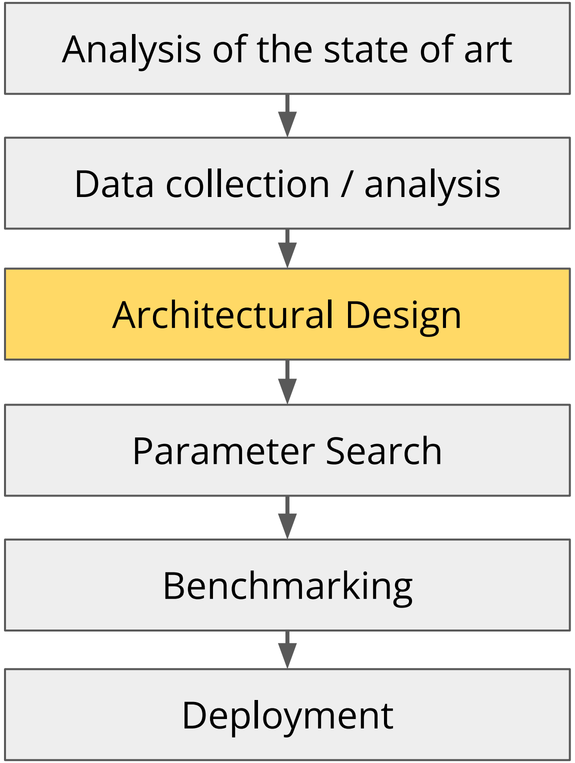
```
# define percentage of splitting
train_perc = 0.8
val_perc = 0.1
# test_perc will be: 1 - train_perc - val_perc

# split train validation and test
train = df.sample(frac = train_perc, random_state=1)
test = df.drop(train.index).sample(frac = val_perc/(1-train_perc), random_state=1)
val = df.drop(train.index).drop(test.index)
```



$$(1-0.8)x = 0.1 \qquad x = \frac{0.1}{1-0.8}$$

R&D process



Analysis of the state of art

Data collection / analysis

Architectural Design

Parameter Search

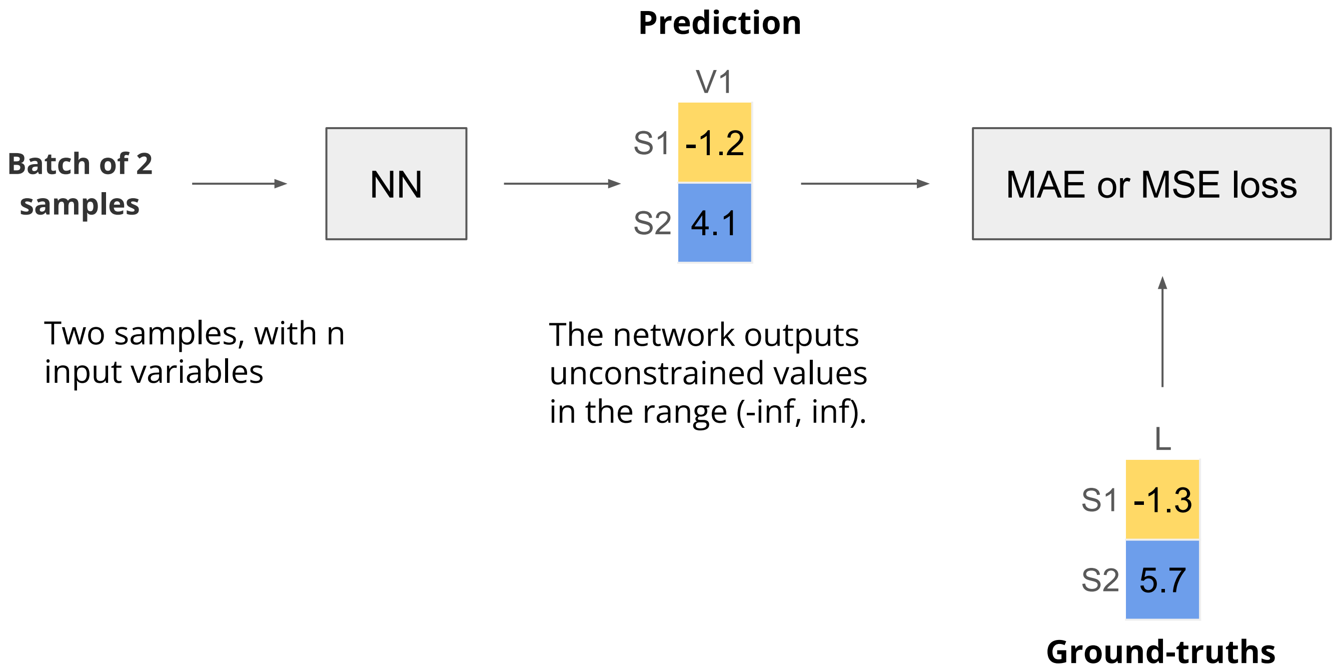
Benchmarking

Deployment

PyTorch PyTorch Lightning

Regression setup

- network predicts directly the values of the continuous variable
- loss and performance score are: MAE or MSE

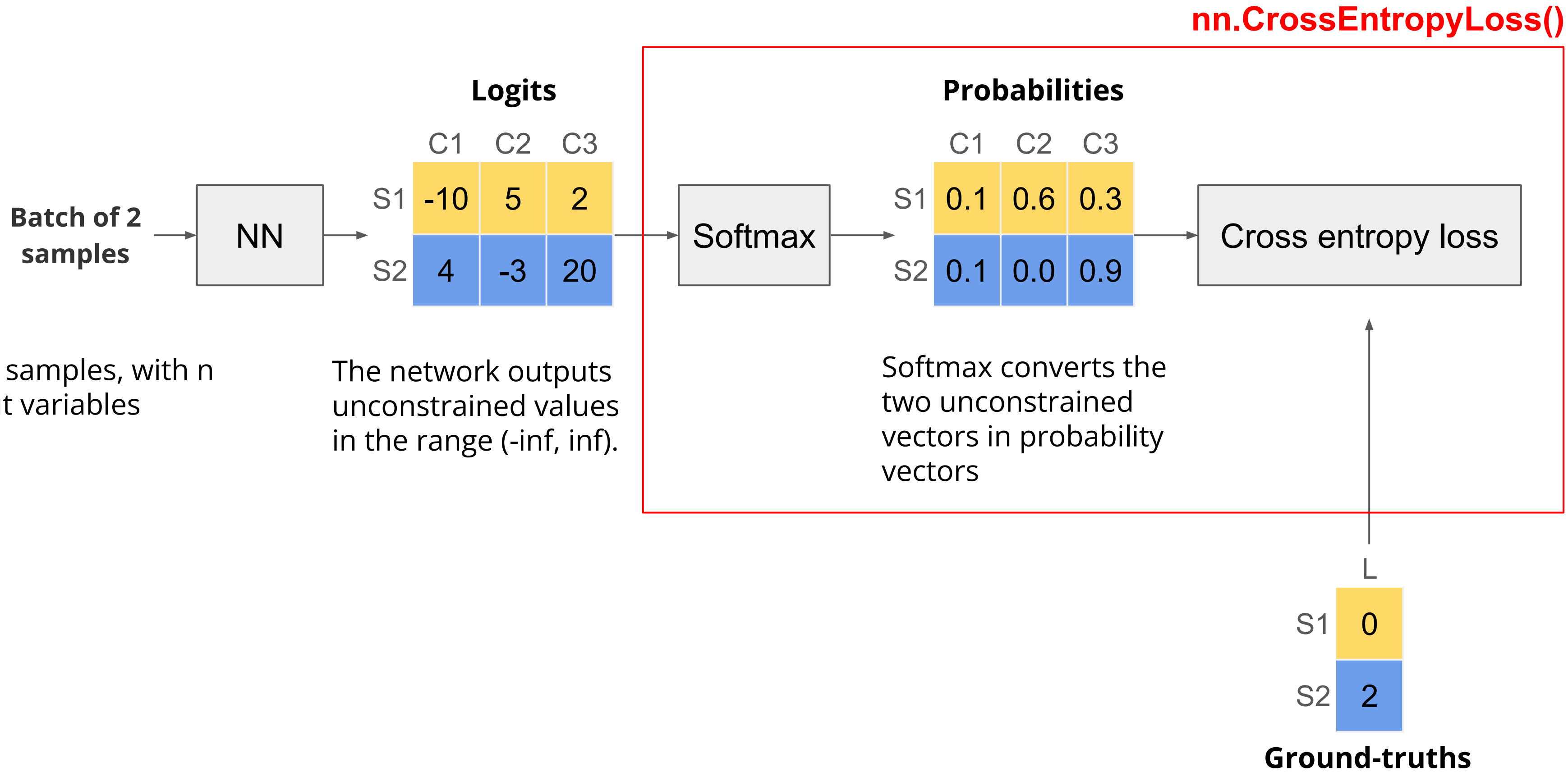


Two samples, with n input variables

The network outputs unconstrained values in the range (-inf, inf).

Pipeline for classification in Pytorch

In Pytorch, nn.CrossEntropyLoss combines softmax and cross entropy loss



Logits

	C1	C2	C3
S1	-10	5	2
S2	4	-3	20

Probabilities

	C1	C2	C3
S1	0.1	0.6	0.3
S2	0.1	0.0	0.9

Ground-truths

S1	0
S2	2

Two samples, with n input variables

The network outputs unconstrained values in the range (-inf, inf).

Softmax converts the two unconstrained vectors in probability vectors

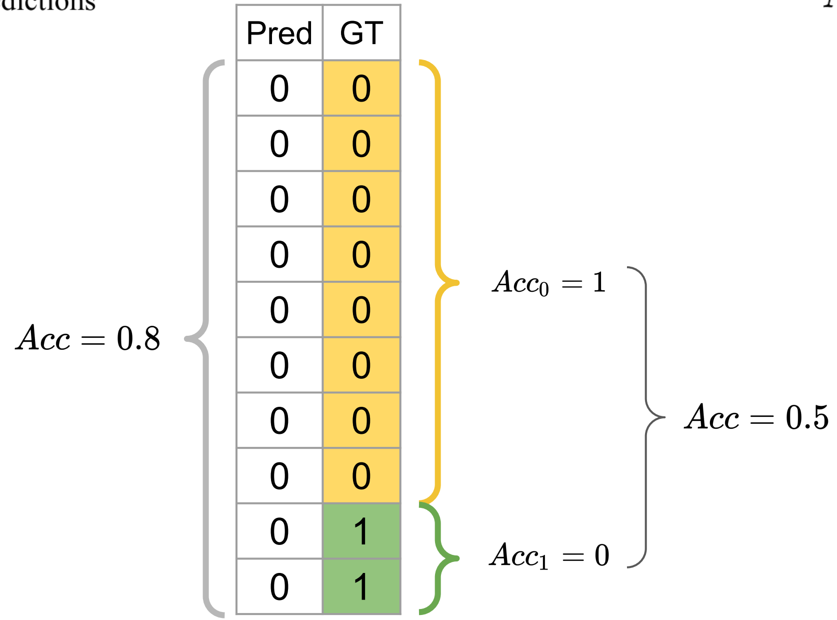
Accuracy

Micro

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Macro

$$\text{Accuracy} = \frac{1}{N}\sum_{c=1}^{N} \frac{\text{\# correct preds for class c}}{\text{\# samples of class c}}$$



Pred	GT
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	1
0	1

$Acc = 0.8$

$Acc_0 = 1$

$Acc_1 = 0$

$Acc = 0.5$

Torchmetrics

```
import torchmetrics

# define input and ground truth
inp = torch.tensor([0,0,0,0,0,0,0,0,0,0])
gt  = torch.tensor([0,0,0,0,0,0,0,0,1,1])
```

```
# define metric objects
acc_micro = torchmetrics.Accuracy(task = 'multiclass', num_classes = 2, average = 'micro')
acc_macro = torchmetrics.Accuracy(task = 'multiclass', num_classes = 2, average = 'macro')
```

Initialization

```
# update metrics
acc_micro.update(inp, gt)
acc_macro.update(inp, gt)

# you can update the metrics with more batches ..
```

Update of the metric.
One update for each batch.

```
# at the end, compute the final score
micro = acc_micro.compute()
macro = acc_macro.compute()
```

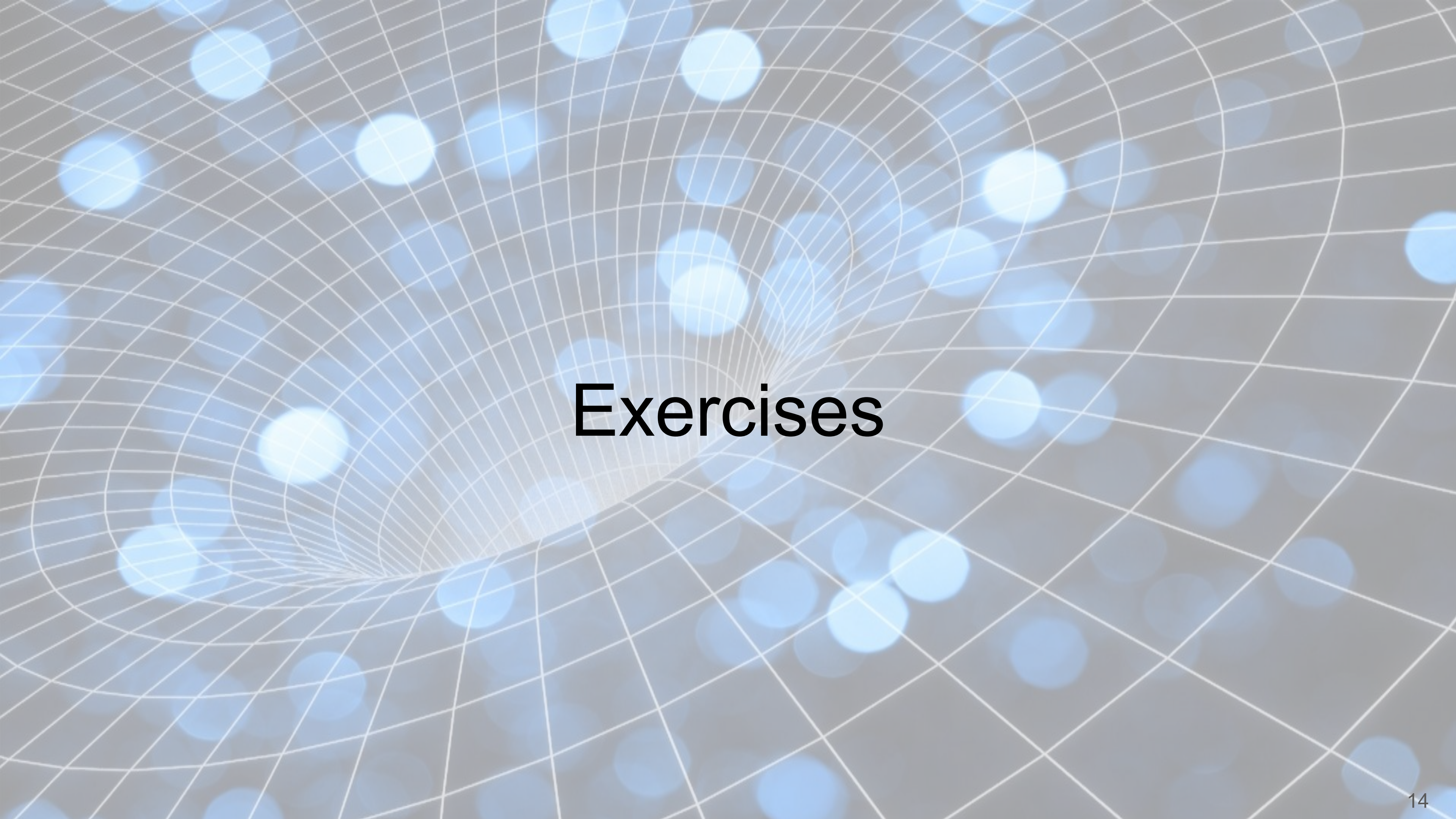
Final computation of the metric

```
# print
print(f'Micro accuracy is {micro:0.2f} while macro accuracy is {macro:0.2f}')
```

```
# reset the metric object (optional)
acc_micro.reset()
acc_macro.reset()
```

Reset of the metric

It will print: “Micro accuracy is 0.80 while macro accuracy is 0.50”



Exercises

Exercise 1 - Data analysis + neural prediction

- Given the dataset “traffic_violations” with target variable “is_arrested”:

1. explore the data
2. decide which variables should be dropped
3. clean data
4. encode data
5. normalize data
6. split data in train, validation and test using the code in slide 8
7. set up the training of a neural network
8. train the system
9. test the performance in terms of micro and macro accuracy
10. compute also the confusion matrix