INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

AI Frameworks

Prédiction de la quantité de pluie journalière

Fethi Benzitouni
Alexandre Bertin
Van Hao Hoang
Hai Vy Nguyen

5th year – Applied Mathematics

7 Janvier 2022
Année 2021/2022

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

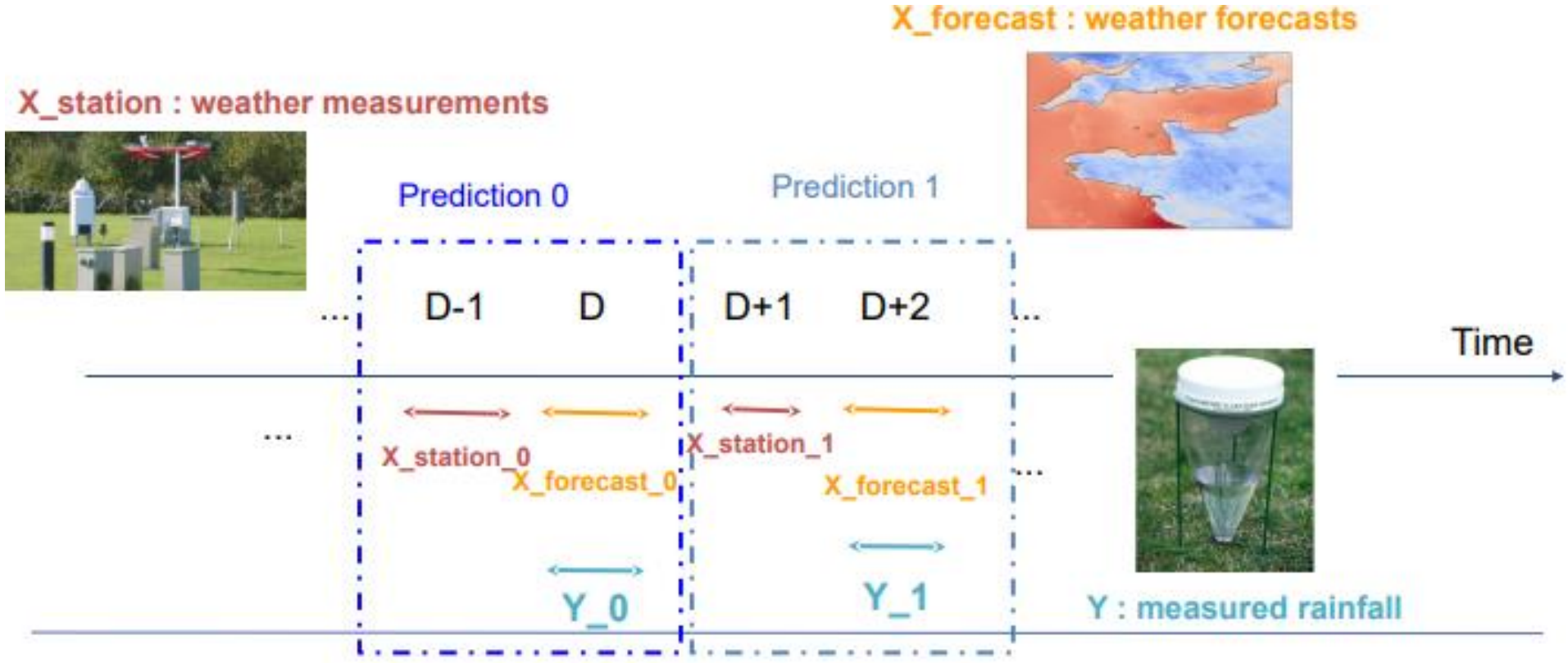
I. Introduction

II. Préparation des données

III. Elaboration de différents modèles

IV. Conclusion

I. Introduction



II. Préparation des données

- **X_station_train: Mesures de différents paramètres de temperature au jour D-1.**
- **2D_arpege_{date}.nc: Mesures de différents paramètres de temperature au jour D. Dans ce cas les données sont calculées et prédites par un modèle physique.**

	number_sta	date	ff	t	td	hu	dd	precip	ld	date_with_hour	ld2
0	14066001	2016-01-01	3.05	279.28	277.97	91.4	200.0	0.0	14066001_0	2016-01-01 00:00:00	14066001_0_0
1	14066001	2016-01-01	2.57	278.76	277.45	91.4	190.0	0.0	14066001_0	2016-01-01 01:00:00	14066001_0_1
2	14066001	2016-01-01	2.26	278.27	277.02	91.7	181.0	0.0	14066001_0	2016-01-01 02:00:00	14066001_0_2
3	14066001	2016-01-01	2.62	277.98	276.95	93.0	159.0	0.0	14066001_0	2016-01-01 03:00:00	14066001_0_3
4	14066001	2016-01-01	2.99	277.32	276.72	95.9	171.0	0.0	14066001_0	2016-01-01 04:00:00	14066001_0_4
...	...	...	...	...	...	...	...	...	...	...	...
4409469	95690001	2017-12-30	9.10	286.68	283.44	80.8	239.0	0.0	95690001_729	2017-12-30 19:00:00	95690001_729_19
4409470	95690001	2017-12-30	8.58	286.39	283.21	81.1	231.0	0.0	95690001_729	2017-12-30 20:00:00	95690001_729_20
4409471	95690001	2017-12-30	8.74	286.28	283.40	82.6	226.0	0.0	95690001_729	2017-12-30 21:00:00	95690001_729_21
4409472	95690001	2017-12-30	9.04	286.21	283.29	82.4	224.0	0.0	95690001_729	2017-12-30 22:00:00	95690001_729_22
4409473	95690001	2017-12-30	9.11	285.92	282.42	79.4	221.0	0.0	95690001_729	2017-12-30 23:00:00	95690001_729_23

4409474 rows × 11 columns

Illustration du fichier X_station_train.csv

▾ Data variables:

ws	(valid_time, latitude, longitude)	float32 ...
p3031	(valid_time, latitude, longitude)	float32 ...
u10	(valid_time, latitude, longitude)	float32 ...
v10	(valid_time, latitude, longitude)	float32 ...
t2m	(valid_time, latitude, longitude)	float32 ...
d2m	(valid_time, latitude, longitude)	float32 ...
r	(valid_time, latitude, longitude)	float32 ...
tp	(valid_time, latitude, longitude)	float32 ...
msl	(valid_time, latitude, longitude)	float32 ...

Illustration du fichier 2D_arpege_{date}.nc

Variable	Missing rate (%)
ff	39.7
t	5.2
td	32.4
hu	32.3
dd	39.7
precip	7.0

Table 1: Missing rate of variables of *X_station_train* file

II. Préparation des données

➢ **À partir de *X_station_train* et *Y_train:***

- Un tableau X_precip de taille (156605, 24)
- Un tableau X_t de taille (156605, 24)
- Un tableau month de taille (156605)
- Un tableau y de taille(156605,)

➢ **À partir de *2D_arpege_{date}:***

- P3031 pression atmosphérique (mean sea level pressure)
- d2m point de rosé (dew point temperature)
- r humidité relative.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

III. Experiment with different models

1. Multi-layer Perception (MLP) model
2. Multi-layer Perception (MLP) mode with Conv1D layer
3. MLP model with fast Fourier transform (FFT)
4. Recurrent neural network model with SimpleRNN and Bidirectional GRU layers
5. Recurrent neural network model with LSTM and Bidirectional GRU layers
6. Recurrent model with Conv1D
7. Uni-dimensional Recurrent Neural Network model
8. Recurrent Model Network trained based on months
9. Recurrent Model Network trained based on groups of months
10. RNN model training on data from the files 2D arpege {date}.nc's

III. Elaboration de différents modèles

1

- Multi-layer Perception (MLP) model

$$X_{s,D-1} = (T_{s,D-1,0}, \ldots, T_{s,D-1,23}, prep_{s,D-1,0}, \ldots, prep_{s,D-1,23})$$ Modèle 1 ⟹ $Y_{s,D}$

Avec :

- La station notée s
- Le jour noté D
- L'heure noté t

Ici, input_shape=48 car concatenation d'une séquence de taille 24 de precip et de celle de t au jour D-1

```
model =km.Sequential()
model.add(kl.Dense(64, input_shape=(48,),  activation = "relu"))
model.add(kl.Dense(128,  activation = "relu"))
model.add(kl.Dropout(0.2))
model.add(kl.Dense(128,  activation = "relu"))
model.add(kl.Dropout(0.2))
model.add(kl.Dense(64,  activation = "relu"))
model.add(kl.Dense(32,  activation = "relu"))
model.add(kl.Dense(1, activation='relu'))
model.compile(loss='mse', optimizer='adam', metrics=['mse'])
model.summary()
```

La structure du modèle

Résultat:
Mean Square Error **(MSE)** sur le jeu d'entraînement: **21.3564**

III. Elaboration de différents modèles

2

- Multi-layer Perception (MLP) mode with Conv1D layer

$$X_{s,D-1,t} = (prep_{s,D-1,t}, T_{s,D-1,t})^T$$

Le problème se modélise de cette façon :
Trouver un modèle tel que :

$$X_{s,D-1} = (X_{s,D-1,0}, \ldots, X_{s,D-1,23})^T \xrightarrow{\text{modèle}} Y_{s,D}$$

Ici, timesteps=24 et input_dim=2 (precip et t)

```
model3 =km.Sequential()
model3.add(kl.Conv1D(32, 3, activation='relu', input_shape=(timesteps, input_dim)))
model3.add(kl.MaxPooling1D(pool_size=3))
model3.add(kl.Flatten())
model3.add(kl.Dense(64, input_shape=(48,),  activation = "relu"))
model3.add(kl.Dense(128,  activation = "relu"))
model3.add(kl.Dropout(0.2))
model3.add(kl.Dense(128,  activation = "relu"))
model3.add(kl.Dropout(0.2))
model3.add(kl.Dense(64,  activation = "relu"))
model3.add(kl.Dense(32,  activation = "relu"))
model3.add(kl.Dense(1, activation='relu'))
```

La structure du modèle:

Le résultat:
Mean Square Error **(MSE)** sur le jeux d'entraînement: **21.6005**
Mean Square Error **(MSE)** sur le jeux de validation: **20.3801**

III. Elaboration de différents modèles

3

- MLP model with fast Fourier transform (FFT)

$$X_{s,D-1} = \{T_{s,D-1,0}, \ldots, T_{s,D-1,23}, prep_{s,D-1,0}, \ldots, prep_{s,D-1,23}, Tf(prep_{s,D-1})[0], .., Tf(prep_{s,D-1})[23], Tf(T_{s,D-1})[0], \ldots, Tf(T_{s,D-1})[23]\}$$

- Centraliser séquence (taille 24) de precip et t
- Effectuer FFT
  => Obtenir 2 nouvelles séquences mais dans le domaine fréquentiel
- Concaterner 4 séquences => input vector de taille 96

```
model4 =km.Sequential()
model4.add(kl.Dense(64, input_shape=(96,),  activation = "relu"))
model4.add(kl.Dense(128,  activation = "relu"))
model4.add(kl.Dropout(0.2))
model4.add(kl.Dense(128,  activation = "relu"))
model4.add(kl.Dropout(0.2))
model4.add(kl.Dense(64,  activation = "relu"))
model4.add(kl.Dense(32,  activation = "relu"))
model4.add(kl.Dense(1, activation='relu'))
opt = tensorflow.keras.optimizers.Adam(learning_rate=1e-3)
model4.compile(loss='mse', optimizer=opt, metrics=['mse'])
```

La structure du modèle

Résultat:
Mean Square Error **(MSE)** sur le jeux d'entraînement: **21.3565**

III. Elaboration de différents modèles

4

- Recurrent neural network model with SimpleRNN and Bidirectional GRU layers

$$X_{s,D-1,t} = (prep_{s,D-1,t}, T_{s,D-1,t})^T$$

Le problème se modélise de cette façon :
Trouver un modèle tel que :

$$X_{s,D-1} = (X_{s,D-1,0}, \ldots, X_{s,D-1,23})^T \Rightarrow Y_{s,D}$$

```
model1 = km.Sequential()
model1.add(kl.SimpleRNN(units=20 ,activation="relu", input_shape=(24, 2),return_sequences=True))
model1.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
model1.add(kl.Dense(1))
model1.summary()
```

La structure du modèle

Le résultat:

Mean Square Error **(MSE)** sur le jeux de validation: **15.0170**.

Le score sur Kaggle est **49.09164**.

III. Elaboration de différents modèles

5

- Recurrent neural network model with LSTM and Bidirectional GRU layers

La structure du modèle

```
model12 = km.Sequential()
model12.add(kl.LSTM(units=20 ,activation="relu", input_shape=(24, 2),return_sequences=True))
model12.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
model12.add(kl.Dense(1))
model12.summary()
```

Le résultat:

Mean Square Error **(MSE)** sur le jeux de validation: **14.8701**

Le score sur Kaggle est **64.56795**

III. Elaboration de différents modèles

6

- Recurrent model with Conv1D

La structure du modèle

```
model3 = km.Sequential()
model3.add(kl.Conv1D(32, 3, activation='relu', input_shape=(24, 2),padding='same'))
model3.add(kl.LSTM(10, return_sequences=True))
model3.add(kl.SimpleRNN(units=10 ,activation="relu"))
model3.add(kl.Dense(1,activation='relu'))
model3.summary()
```

Le résultat:

Mean Square Error **(MSE)** sur le jeux de validation: **21.3564**

III. Elaboration de différents modèles

7

- Uni-dimensional Recurrent Neural Network model

La structure du modèle

```
model3 = km.Sequential()
model3.add(kl.Dense(units=48 ,activation="relu", input_shape=(24,)))
model3.add(kl.Dense(units=24 ,activation="relu"))
model3.add(kl.Reshape((24,1),input_shape=(24,)))
model3.add(kl.GRU(units=20 ,activation="relu", input_shape=(24, 1),return_sequences=True))
model3.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
model3.add(kl.Dense(units=20 ,activation="relu"))
model3.add(kl.Dense(1))
model3.summary()
```

Le résultat:

Mean Square Error **(MSE)** sur le jeux de validation: **14.8847**

Le score sur Kaggle est **46.78**

Modification apporté : élimination des outliers , entraînement sur log(Y) (MSE)

III. Elaboration de différents modèles

8

- Recurrent Model Network trained based on months

```
def make():
  model = km.Sequential()
  model.add(kl.SimpleRNN(units=20 ,activation="relu", input_shape=(24, 2),return_sequences=True))
  model.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
  model.add(kl.Dense(1))
  model.compile(loss="mse", optimizer="adam")
  return model
```

La structure du modèle

Le résultat:
Le score sur Kaggle est **86.212**

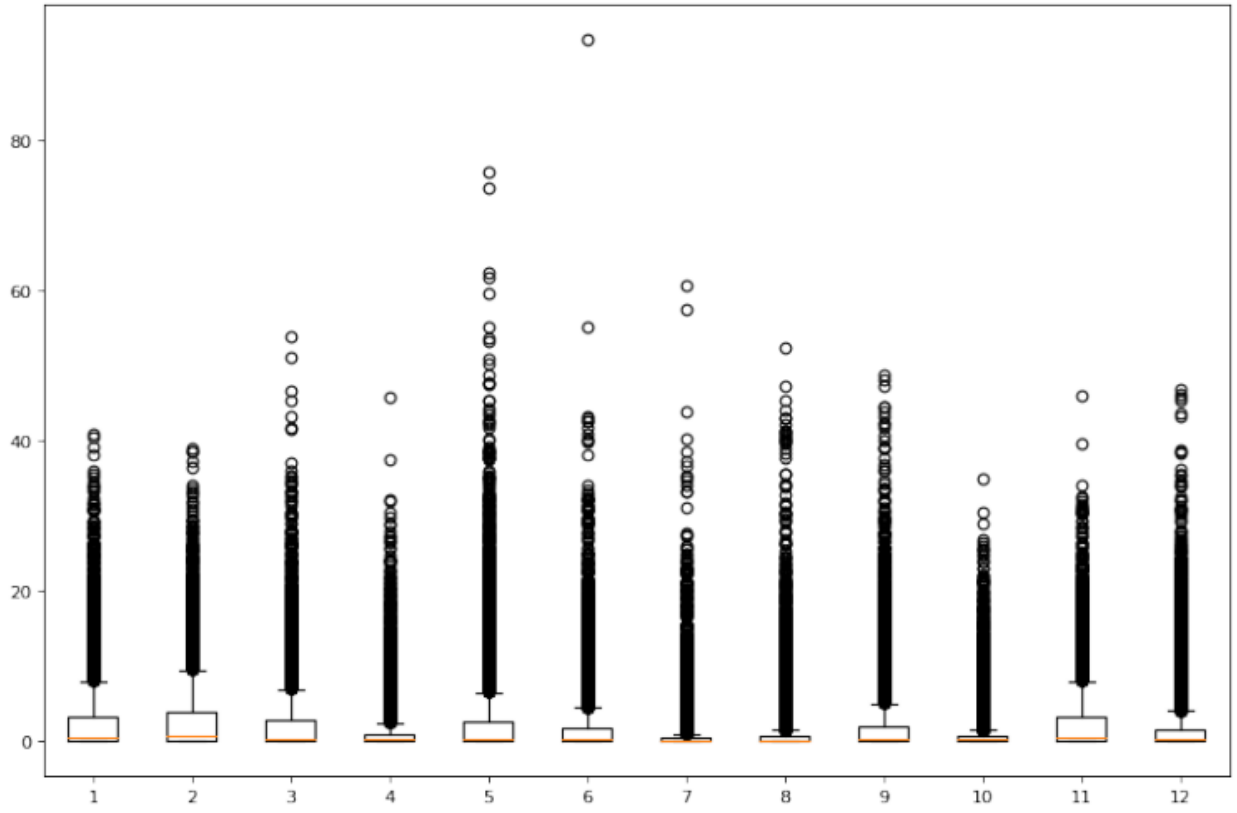
III. Elaboration de différents modèles

9

- Recurrent Model Network trained based on groups of months

Sur la base de la forme de boxplots, nous pouvons diviser les 12 mois en groupes qui ont une distribution similaire des précipitations quotidiennes :

- **Groupe 1 : avril, juillet, août, octobre**
- **Groupe 2 : juin, septembre, décembre**
- **Groupe 3 : janvier, février, mars, mai, novembre**



Boxplot de Y pour 12 mois

9. Recurrent Model Network trained based on groups of months

III. Elaboration de différents modèles

9

- Recurrent Model Network trained based on groups of months

```
def make_group():
  model_group = km.Sequential()
  model_group.add(kl.Dense(units=48 ,activation="relu", input_shape=(24,)))
  model_group.add(kl.Dense(units=24 ,activation="relu"))
  model_group.add(kl.Reshape((24,1),input_shape=(24,)))
  model_group.add(kl.GRU(units=20 ,activation="relu", input_shape=(24, 1),return_sequences=True))
  model_group.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
  model_group.add(kl.Dense(units=20 ,activation="relu"))
  model_group.add(kl.Dense(1))
  return model_group
```

La structure du modèle

Le résultat:

Le score sur Kaggle est **54.243**

III. Elaboration de différents modèles

10

- RNN model training on data from the files 2D arpege {date}.nc's

p3031 (mean sea level pressure),**d2m** (dew point temperature) **et r** (relative humidity)

```
model3 = km.Sequential()
model3.add(kl.Dense(units=48 ,activation="relu", input_shape=(24,)))
model3.add(kl.Dense(units=24 ,activation="relu"))
model3.add(kl.Reshape((24,1),input_shape=(24,)))
model3.add(kl.GRU(units=20 ,activation="relu", input_shape=(24, 1),return_sequences=True))
model3.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
model3.add(kl.Dense(units=20 ,activation="relu"))
model3.add(kl.Dense(1))
model3.summary()
```

Entrainer le réseau

Si MSE > 17 => Rejeter

r et p3031

	number_sta	lat	lon	height_sta	lat2	lat_index	lon2	lon_index
0	86118001	46.477	0.985	120.0	46.495998	54	0.958	68
1	86149001	46.917	0.025	60.0	46.896000	50	0.058	59
2	56081003	48.050	-3.660	165.0	48.096001	38	-3.642	22
3	53215001	47.790	-0.710	63.0	47.796001	41	-0.742	51
4	22135001	48.550	-3.380	148.0	48.596001	33	-3.342	25
...	...	...	...	...	...	...	...	...
320	86137003	47.035	0.098	96.0	46.995998	49	0.058	59
321	86165005	46.412	0.841	153.0	46.396000	55	0.858	67
322	86273001	46.464	1.042	121.0	46.495998	54	1.058	69
323	91200002	48.526	1.993	116.0	48.495998	34	1.958	78
324	95690001	49.108	1.831	126.0	49.096001	28	1.858	77

325 rows × 8 columns

Chercher le point sur la grille le plus proche de chaque station

III. Elaboration de différents modèles

10

- RNN model training on data from the files 2D arpege {date}.nc's

Nous avons approché la location de toutes les stations à un point de la grille pour obtenir et utiliser un algorithme pour conserver les meilleurs paramètres ainsi :

$$X_{s,D,t} = (p3031_{s,D,t}, r_{s,D,t})^T$$

Le problème se modélise de cette façon :

Trouver un modèle tel que :

$$X_{s,D} = (X_{s,D,0}, \ldots, X_{s,D,23})^T \xrightarrow{\text{modèle}} Y_{s,D}$$

```
model2 = km.Sequential()
model2.add(kl.GRU(units=20 ,activation="relu", input_shape=(24, 2),return_sequences=True))
model2.add(kl.Bidirectional(kl.GRU(units=20 ,activation="relu")))
model2.add(kl.Dense(units=20 ,activation="relu"))
model2.add(kl.Dense(1))
model2.summary()
```

La structure du modèle

Le résultat:

Mean Square Error **(MSE)** sur le jeux de validation: **13.7765<14.**

Le score sur Kaggle est **83.28281**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

IV. Conclusion

- Modèles construit autour des technologies MLP et RNN.
- Commencement avec un modèle MLP.
- Développement de modèles RNN.
- Utilisation des techniques FFT, ondelettes et conv1D.
- Utilisation des fôrets aléatoires.
- Le modèle RNN uni-dimensionnel donne le meilleur score.
- Basée sur MSE, le dernier modèle donne de meilleurs résultats.
- Sans données manquantes nous aurions de meilleurs résultats.

Merci pour votre attention