P3

April 30, 2021

1 MA8701 Project II (Neural Nets)

A comprehensive introduction to the data set can be found in the `Project2.Rmd` in the same repository.

(TL;DR: The data set is available at https://github.com/metno/havvarsel-data-driven-pred/blob/main/dataset1.csv)

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
# Make numpy printouts easier to read.
np.set_printoptions(precision=3, suppress=True)
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
# print(tf.__version__)
```

```
INFO:tensorflow:Enabling eager execution
INFO:tensorflow:Enabling v2 tensorshape
INFO:tensorflow:Enabling resource variables
INFO:tensorflow:Enabling tensor equality
INFO:tensorflow:Enabling control flow v2
```

1.0.1 Preprocessing

The data is preprocessed the same way as in the `Project2.Rmd` (i.e., the response is constructed as the `water_temp` at the next time point and the last year is set apart as test data)

[3]:
```
# Loading data
raw_dataset = pd.read_csv('dataset1.csv', header = 0)
# Constructing response
shifted_water_temp = raw_dataset['water_temp']
shifted_water_temp = shifted_water_temp.shift(1)
dataset = raw_dataset
dataset['response'] = shifted_water_temp
dataset = dataset.iloc[1:, :]
```

```
dataset = dataset.set_index("time")
del dataset["SN19940wind_speed1"] # remove empty data
del dataset["SN18700cloud_area_fraction1"] # remove empty data

# Casting to entries to floats
for i in range(len(dataset.columns)):
    dataset.iloc[:,i] = pd.to_numeric(dataset.iloc[:,i], downcast = "float")

dataset.head()
```

[3]:

time	water_temp	SN19710air_temperature0	SN19923air_temperature0	SN19430air_temperature0	SN19940air_temperature0	SN19923wind_speed0	SN19940wind_speed0	SN18700cloud_area_fraction0	SN17150cloud_area_fraction0
2016-05-24 10:00:00+00:00	12.0	9.8	9.0	11.700000	11.1	3.6	1.4	8.0	8.0
2016-05-24 14:00:00+00:00	12.1	13.0	12.7	14.500000	14.2	4.2	1.4	7.0	
2016-05-24 16:00:00+00:00	12.3	12.4	12.5	14.500000	13.9	5.0	1.5	7.0	
2016-05-25 07:00:00+00:00	11.6	11.9	10.0	13.100000	11.6	3.4	1.8	1.0	
2016-05-25 11:00:00+00:00	12.6	15.4	15.0	17.200001	16.9	3.9	1.4	1.0	

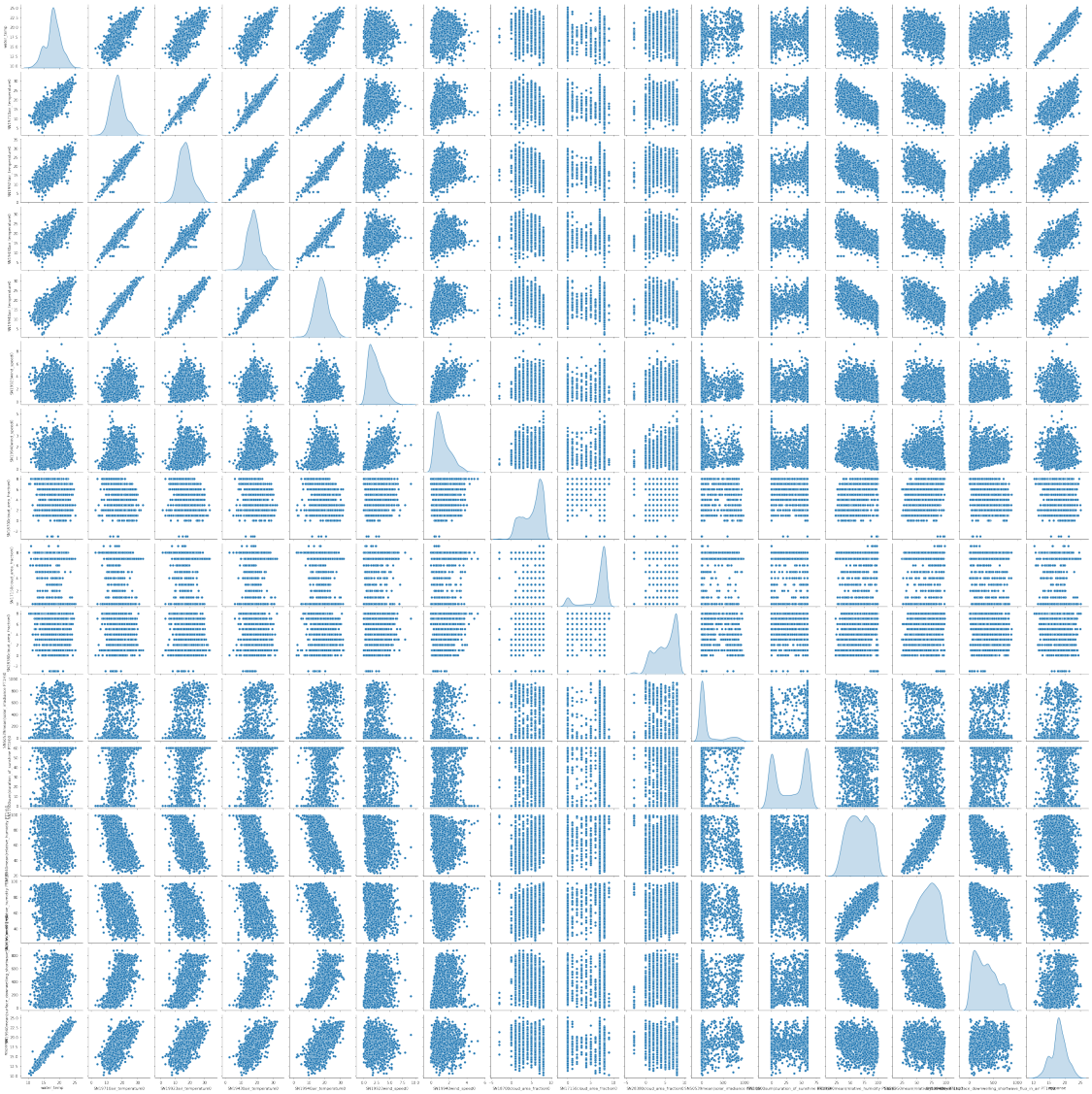
```
2016-05-24 14:00:00+00:00                          8.0
2016-05-24 16:00:00+00:00                          8.0
2016-05-25 07:00:00+00:00                          8.0
2016-05-25 11:00:00+00:00                          8.0

                          SN28380cloud_area_fraction0  \
time
2016-05-24 10:00:00+00:00                          5.0
2016-05-24 14:00:00+00:00                          5.0
2016-05-24 16:00:00+00:00                          3.0
2016-05-25 07:00:00+00:00                          4.0
2016-05-25 11:00:00+00:00                          1.0

                          SN50539mean(solar_irradiance PT1H)0  \
time
2016-05-24 10:00:00+00:00                                0.737
2016-05-24 14:00:00+00:00                                0.737
2016-05-24 16:00:00+00:00                                0.737
2016-05-25 07:00:00+00:00                                0.737
2016-05-25 11:00:00+00:00                                0.737

                          SN11500sum(duration_of_sunshine PT1H)0  \
time
2016-05-24 10:00:00+00:00                                   0.0
2016-05-24 14:00:00+00:00                                   0.0
2016-05-24 16:00:00+00:00                                   1.0
2016-05-25 07:00:00+00:00                                   0.0
2016-05-25 11:00:00+00:00                                  60.0

                          SN19940mean(relative_humidity PT1H)0  \
time
2016-05-24 10:00:00+00:00                                  83.0
2016-05-24 14:00:00+00:00                                  66.0
2016-05-24 16:00:00+00:00                                  64.0
2016-05-25 07:00:00+00:00                                  59.0
2016-05-25 11:00:00+00:00                                  48.0

                          SN26950mean(relative_humidity PT1H)0  \
time
2016-05-24 10:00:00+00:00                                  76.0
2016-05-24 14:00:00+00:00                                  78.0
2016-05-24 16:00:00+00:00                                  69.0
2016-05-25 07:00:00+00:00                                  55.0
2016-05-25 11:00:00+00:00                                  48.0

                          SN19940mean(surface_downwelling_shortwave_flux_in_air
PT1H)0  \
```

```
time
2016-05-24 10:00:00+00:00                                    200.899994
2016-05-24 14:00:00+00:00                                    330.000000
2016-05-24 16:00:00+00:00                                    281.600006
2016-05-25 07:00:00+00:00                                    392.799988
2016-05-25 11:00:00+00:00                                    801.000000

                           response
time
2016-05-24 10:00:00+00:00      11.7
2016-05-24 14:00:00+00:00      12.0
2016-05-24 16:00:00+00:00      12.1
2016-05-25 07:00:00+00:00      12.3
2016-05-25 11:00:00+00:00      11.6
```

[4]:
```
# Split the data set
train_dataset = dataset.iloc[:1856, :] # split the data in 2016-2019 into␣
↪training data
test_dataset = dataset.drop(train_dataset.index) # use the data in 2020 as the␣
↪test data
```

1.0.2 Data exploration (continuation)

[5]:
```
sns.pairplot(train_dataset, vars = train_dataset.columns[:], diag_kind='kde')
plt.show()
```



One can tell that the air temperature and water temperature tend to have very high correlations regardless of their measurement stations, which makes sense since the air temperature will mostly likely be pretty similar when they are measured in a relatively close region.

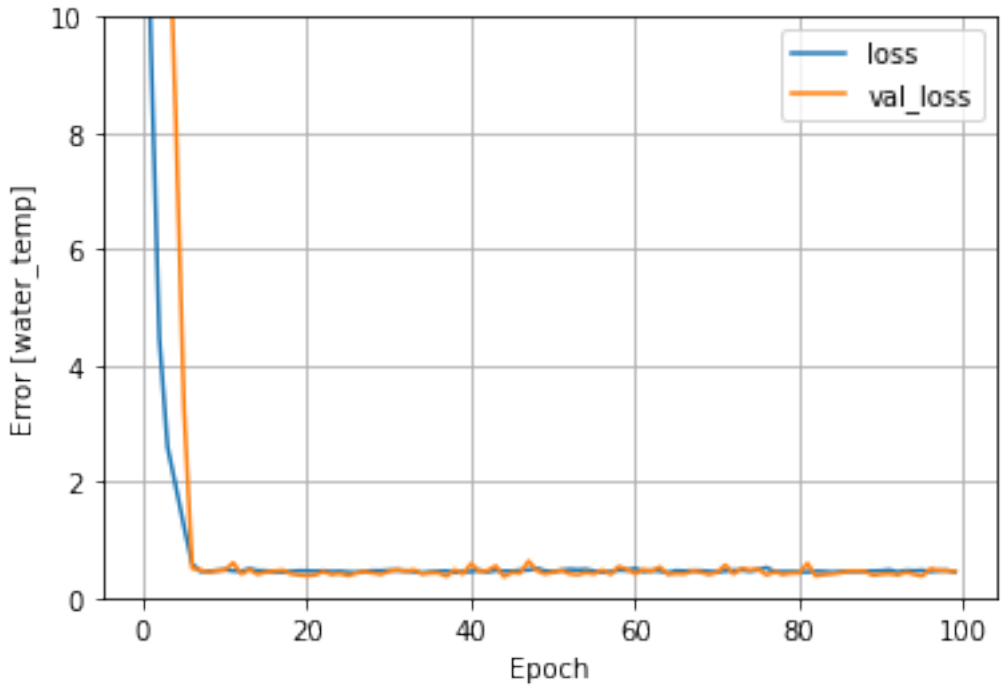
1.1 Neural Nets

Due to complexity restrictions, only one neural network is going to be trained and tested here.

[6]:
```
train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('response')
test_labels = test_features.pop('response')
```

[7]:
```
normalizer = preprocessing.Normalization()
normalizer.adapt(np.array(train_features))

def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 10])
  plt.xlabel('Epoch')
  plt.ylabel('Error [water_temp]')
  plt.legend()
  plt.grid(True)
```

[8]:
```
linear_model = tf.keras.Sequential([
    normalizer,
    layers.Dense(units=1)
])

linear_model.predict(train_features)
linear_model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.1),
    loss='mean_absolute_error')

history = linear_model.fit(
    train_features, train_labels,
    epochs=100,
    # suppress logging
    verbose=0,
    # Calculate validation results on 20% of the training data
    validation_split = 0.2)

plot_loss(history)
```



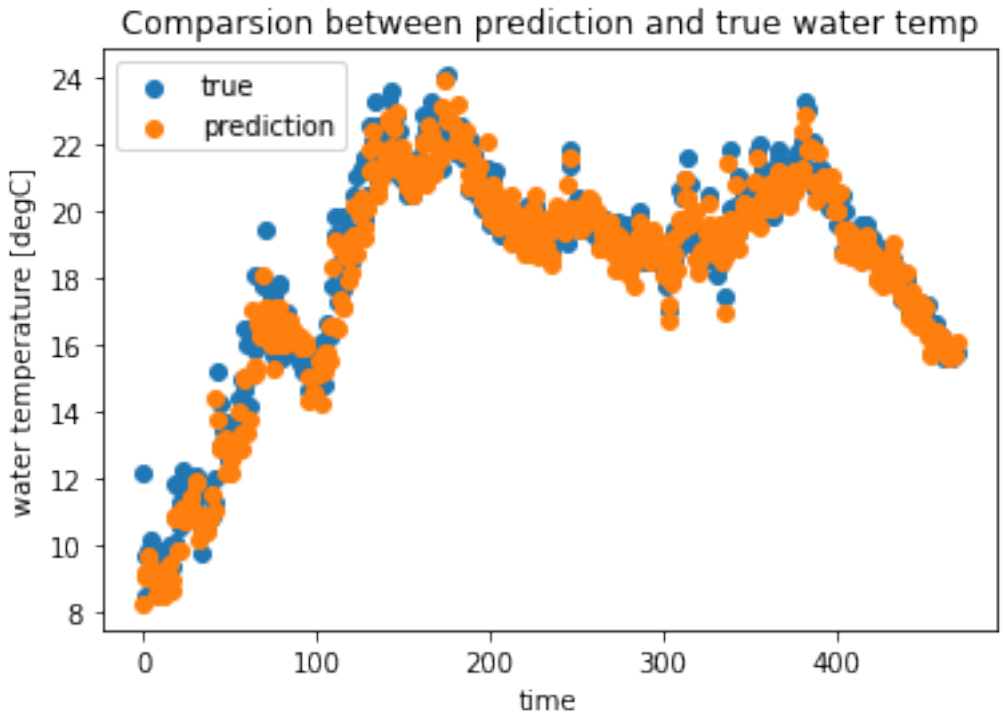
The training history plot yields that the training is pretty fast to converge - the loss is minimized quickly. This is one the of the benefit for using the simple model. `Loss` represents the loss during the training while `val_loss` represents the loss for the validation set during the training.

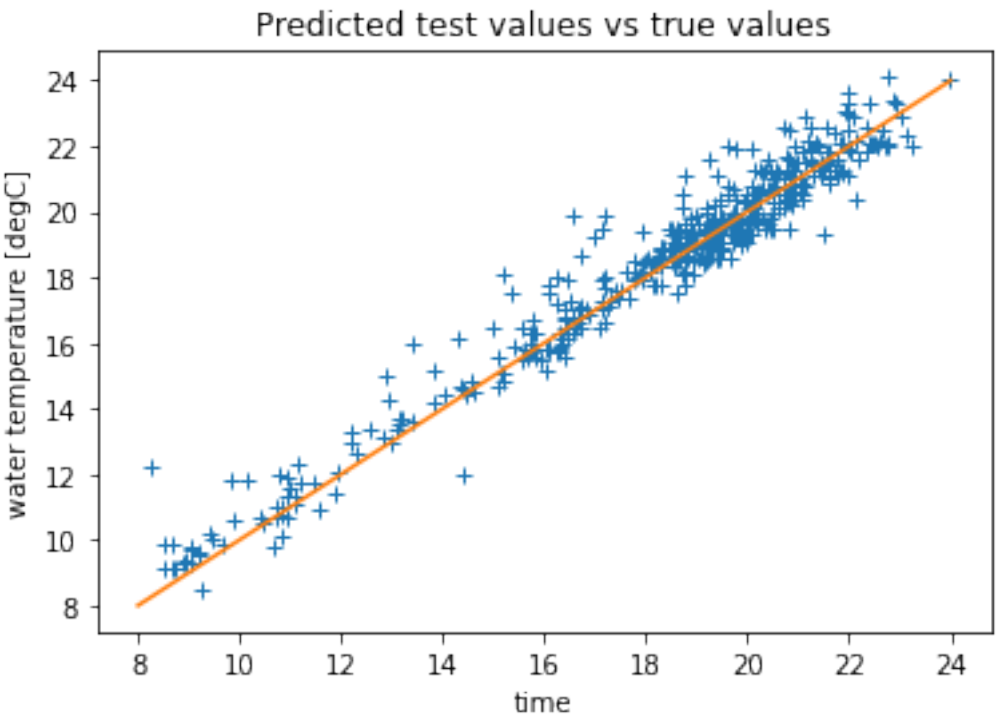
1.1.1 Prediction

We evaluate the performance of the trained neural net on the test set which are the observations from the last year.

[9]:
```
test_results = linear_model.evaluate(
    test_features, test_labels, verbose=0)
test_predictions = linear_model.predict(test_features).flatten()
# plot the estimation
x = np.arange(len(test_labels))
plt.scatter(x, test_labels)
plt.scatter(x, test_predictions)
plt.title("Comparsion between prediction and true water temp")
plt.legend(['true', 'prediction'])
plt.ylabel("water temperature [degC]")
plt.xlabel("time")
plt.show()

plt.plot(test_predictions, test_labels, "+")
plt.plot(np.linspace(8,24), np.linspace(8,24))
plt.title("Predicted test values vs true values")
```

```
plt.xlabel("time")
plt.ylabel("water temperature [degC]")
plt.show()

MSE = linear_model.evaluate(test_features, test_labels, verbose=0)

print("The MSE for the prediction sets is ", MSE)
```






```
The MSE for the prediction sets is  0.5626466870307922
```

The trained neural net shows a good performance on the test set. The prediction seems to be unbiased and with low variance.

2 Conclusion

We have fetched a dataset from different sources, which contains the water temperature at a popular swimming site in Asker and collected atmospheric measurements from the MET weather observation stations around. The goal of our project work was to do prediction for the water temperature of the next future time using all information of the current time. The data exploration suggests the need for non-linear data analysis method, wherefore we have fit a random forest model and a neural net. Both approaches show a grandious prediction quality. The random forest is a bit biased for low temperatures, but in contrast to the neural net it allows us to identify the most significant variables, which coincide qulitatively to a large extend with our intuition.