Machine Learning project CS-C3240, Spring 2022

Simplistic weather model

In this project I have decided to implement a simplified weather prediction model. By using freely available weather measurement data one can try to predict the following day's minimum and maximum temperature. One can use this model to forecast the following day's temperature and use that information however sees fit.

Implementation

In this project I've used weather data obtained from the Finnish Meteorological Institute. [2] Every datapoint holds weather measurements from a whole day. It includes timestamps, air temperature, wind speed, precipitation, atmospheric pressure, and more. For a comprehensive weather model one would use all possible data. For this project I've decided to, - somewhat arbitrarily, use the month and day, and air temperature and precipitation as the features. The month number correlates with season which will dictate the general fluctuation of temperature. The same air temperature measurements from a winter afternoon and summer morning will produce different predictions for the next day. Water has a high specific heat capacity which implies that two days with the same air temperature measurements will likely have different predictions for the next day, that is, rain will cool the air down. For this reason, precipitation is included. Air pressure is an indication of how likely it is to rain, therefore it is also included in the list of features.
There are six measurements of air temperature and precipitation spaced 4 hours apart throughout a day. The labels chosen are the following day's minimum and maximum temperatures. The raw data spans 4030 days of measurements. All data is numerical, which will influence the choice of loss functions later.

Several models have been chosen and their results have been compared with each other. Weather in itself has so many different aspects influencing it, which makes predicting weather difficult. I would have chosen a statistical model as that inherently deals with randomness. However, such methods are not introduced in this course, so I've chosen two alternative models, namely polynomial regression and multilayer perceptrons (MLP for short). These have the ability to capture some of the noisy and possibly non-linear behaviour of weather patterns.

Two loss functions have been chosen for polynomial regression, - mean squared error (MSE for short) and Huber loss. MSE has the advantage of being computationally efficient. It also captures outliers, which in this case may be beneficial since weather can fluctuate unexpectedly. Huber loss being more robust against outliers, compared to MSE that is, allows it to act as a comparison. Since all data is numerical, loss functions such as logistical loss, hinge loss, and others that deal with binary classification are excluded from consideration.

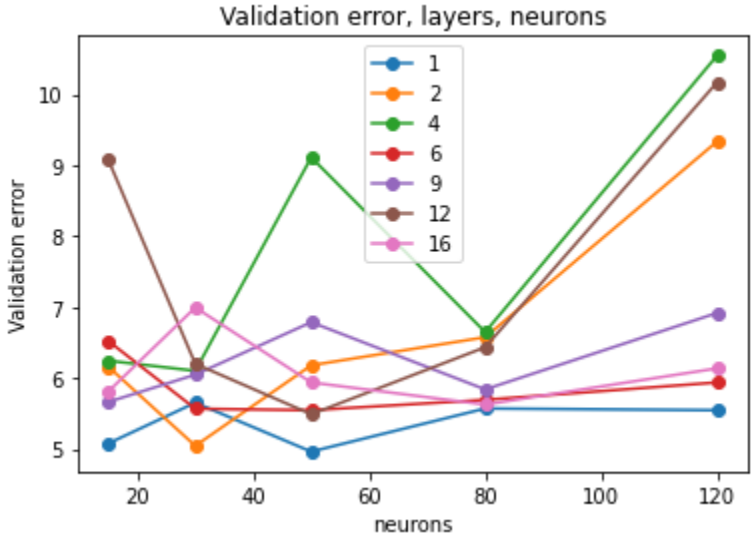
The data has been randomly split into three sets, - a training set, a validation set, and a test set. The size ratio of the training and validation sets is 5:4 by the criteria that the former must be larger than the latter, but not too large (e.g. 9:1) to dwarf it. This is also to prevent overfitting. The dataset with n = 4030 is large enough that I've decided not augment it further.

Results

For all methods their training and validation errors are calculated using the mean squared error to allow for comparison against each other. Both errors for polynomial regression are:

Degree of polynomial	Polynomial regression, MSE		Polynomial regression, Huber	
	Training error	Validation error	Training error	Validation error
1	4.06545762	4.05745078	4.19024753	4.21648787
2	3.12604270	4.55985463	4.12503779	4.10957624
3	0.925133817	274033.589	4.14329055	4.09798493
4			4.11577842	4.0491982

Here one can observe that by degree 3 polynomial regression with MSE is grossly overfitting the model. The most one can use is degree. In contrast, when using Huber loss the model seems to not have reached optimal training yet. More options could be explored with Huber loss, such as setting specific values for epsilon, the variable indicating the threshold for outlier values. [1 pages 84-85] During all stages of training, it was left at its default value of 1.35 . [3]



With MLP as the model, the number of neurons and layers both have an influence in the errors. Multiple setups have been tested, and their errors can be seen from the appendix, most of their values lie between [5.0, 7.0] for training and validation errors. A graph of only validation errors can be seen in the graph on the right. Each line indicates how many layers are in the model.

An interesting note here: In my earlier dataset there were 769 datapoints only, and for such a set even a degree = 2 polynomial model with MSE was overfitting the data with test error < 1 and validation error 10 < v_err < 100.

After a direct comparison using the lowest validation error as the sole criterion, the model using polynomial regression degree 4 and Huber loss is chosen. It has validation error 4.0491. Applying the test data set on the model it yields 4.2261 as the test error.

Conclusion

The goal of this project has been to construct a simplistic weather model using a few selected types of weather measurements. Its only function is to take one day's measurements and predict the minimum and maximum temperature of the following day. After retrieving and processing the data from the Finnish Meteorological Institute's website, it has been used as training data for two types of models: Polynomial regression and multilayer perceptron. When using mean squared error as the loss function for polynomial regression, the model has reached an optimal complexity at degrees 1 and 2, while using Huber loss indicates there still is room for improvement. All multilayer perceptron models have shown higher validation errors than polynomial models, therefore other types of models are prioritised.

With the chosen polynomial model with Huber loss, improvement can be made, such as tuning its threshold parameter for outlier datapoints. A disadvantage of the selected model is its high complexity of degree 4. Training such a model takes noticeably longer time than other simpler models: 10 minutes for degree 4 vs. 1 minute for degree 3. This encourages reducing model complexity and using other methods to improve the model. Principle component analysis would be one option. Using it would allow using bigger datasets without introducing irrelevant features into the training process.

Sources:
[1] https://github.com/alexjungaalto/MachineLearningTheBasics/blob/master/MLBasicsBook.pdf
[2] https://en.ilmatieteenlaitos.fi/download-observations , Kaisaniemi weather station, data from the period 2011.1.1 - 2022.2.28
[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.HuberRegressor.html?highlight=huber#sklearn.linear_model.HuberRegressor

Source code (Python) appended at the end