Machine Learning
Università della Svizzera italiana

Assignment 1

Barreiro Pérez Diego

May 9, 2022

This assignment is split into two parts: solving a regression problem, and answering some questions.

1 Regression problem

This section of the assignment has been developed in the `src/` folder. Two new files, `file1.py` and `file2.py` have been created for *Task 1* and *Task 2*, respectively.
In these files, some `.pickle` models have been created, which can be later evaluated at `run_model.py`. These models have the 100% set of training.

1.1 Task 1

For *Task 1*, it was asked to use the family of models $f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot \cos(x_2) + \theta_4 \cdot x_1^2$ to fit the data.
Thus, to accomplish this task, the `sklearn.linear_model` was used, specifically the `LinearRegression` model. Data was loaded using the provided function, but it had missing the "$x_3 = cos(x_2)$" and "$x_4 = x_1^2$". So, a loop was used to calculate the respective $cos(x_2)$ and x_1^2 for each data entry.
The next step was to generate the X matrix. However, in the case of θ_0, it had a constant value, so it could be omitted (and treated as the y intercept value). Then, x_3 and x_4 values were appended vertically to the matrix.
Finally, this X matrix can be sent to the model to calculate the intercept (θ_0) and the coefficients ($theta_1$ to $theta_4$). The result is the following one:

$$f(\mathbf{x}) = 1.70 - 0.14 \cdot x_1 - 0.62 \cdot x_2 + 0.02 \cdot \cos(x_2) + 0.04 \cdot x_1^2$$

However, to evaluate the performance, it was not used the entire training dataset. Despite being provided with the training one (as the testing is hidden), to properly evaluate the model, it was split into 80% training and 20% testing. As it can be seen in the upcoming results, it did not perform very well:

- Mean-Square Error: 1.4104
- R-Square: 0.2811
- Variance: 5.6388

1.2 Task 2

For this second task, several regressors were available. In particular, four of them were tested:

- K-Neighbours Regressor: `sklearn.neighbors.KNeighborsRegressor`
- Decision Tree Regressor: `sklearn.tree.DecisionTreeRegressor`
- Support Vector Regressor: `sklearn.svm.SVR`
- Random Forest Regressor: `sklearn.ensemble.RandomForestRegressor`

In the end, the **Support Vector Regressor** was chosen as it was more customizable, it had an excellent generalization capability and performed better than the other models. Dataset does not need any modification (not as in linear regression), as no "extra" variables were needed. So, after invoking the code and adjusting the parameters, the following values were obtained (using all default parameters except gamma="auto" and C=100):

- Mean-Square Error: 0.0127
- R-Square: 0.9935
- Variance: 0.0003

As it can be clearly seen, it performed statistically better than the linear model. Again, these values are obtained from splitting the dataset into the 80%-20% proportions. The saved pickle model has all the training dataset.

1.3 Task 3 (Bonus)

In *Task 2*, **SVR** was the chosen regressor. It had an MSE of 0.0105 over the 100% of the training set, which is better for both training and testing sets with the baseline model.
It can work better using this regressor rather than the baseline as data may be non-linear. **Support Vectors** work well when data is unknown, as it can be generalized and its values can be easily updated.

2 Questions

2.1 Q1. Training versus Validation

1.Q. Explain the behaviors of the curves in each of the three highlighted sections in the figure, namely (a), (b), and (c);

1.A. In the graph we can see how the *Expected Test Error*, *Observed Validation Error* and the *Observed Training Error* change depending on the *Model Complexity* and the *Performance*.
In section **(a)**, as it has a very low *Model Complexity*, all the errors will be high, as the model does not have enough data to properly regress the values.
However, later on, in section **(b)**, *Model Complexity* increases, producing better results with lower errors.
But if this *Model Complexity* keeps increasing as in section **(c)**, the model can be over-fitted, causing a very low error on the *Observed Training Error* (as the model has been trained "very well" with this training set), but higher errors for the *Observed Validation Error*.

2.Q. Do you think the figure gives useful information to reduce the approximation risk? And to reduce the estimation risk? Explain why.

2.A. The approximation risk depends only on how close the approximating model family is to the process generating the data. This means that in the figure, a lower approximation risk could be appreciated when the *Observed Validation Error* stops decreasing earlier (as it would mean that our model is good for the specific data).
The estimation risk depends on the effectiveness of the learning procedure. So, a lower estimation risk can be appreciated when there is a bigger number of optimal models (in the graph, it can be seen when there is a big distance between the *Observed Validation Error* increase and decrease, a "valley").

3.Q. Would you think that by further increasing the model complexity you would be able to bring the structural risk to zero? How would your answer change if your data was not affected by noise?

3.A. No, it is not possible. The structural risk is the generalization ability, the performance at task. Despite having good results with the *Training Set*, its performance decreases with the *Test Set*, increasing the structural risk.

4.Q. If the X axis represented the training iterations instead, would you think that the training procedure that generated the figure used early stopping? Explain why. (**NB:** ignore the subfigures and the dashed vertical lines)

4.A. No, as training should have stopped after detecting an increase in error, which did not happen.

2.2 Q2. Linear Regression

Comment and compare how the (a.) training error, (b.) test error and (c.) coefficients would change in the following cases:

1.Q. $x_3 = x_1 + 3.0 \cdot x_2$.

1.A. As x_3 is a linear combination of x_1 and x_2, which are already included in the calculation, it means that both training error and test error will remain the same. However, coefficients will be different now.

2.Q. $x_3 = x_1 \cdot x_2 \cdot x_2$.

2.A. In this case, x_3 is not a linear combination, meaning both errors and combinations will be different. However, as the set is unknown, it is not possible to determine the change in these values.

3.Q. Can we make any educated guess on what would be the value of θ_3 for each of the preceding cases if we used Lasso Regression?

3.A. The Lasso Regression, in contrast to Ridge, will tend to force some coefficients to 0 if they are not "important" for the model, i.e. if they are dependent.
So, for $x_3 = x_1 + 3.0 \cdot x_2$, as it is lineally dependant, it will tend to be 0.
But for $x_3 = x_1 \cdot x_2 \cdot x_2$ it is not possible to make any assumption without having full knowledge of the data (but presumably it will not be 0 as it is not lineally dependant).

4.Q. Explain the motivation behind Ridge and Lasso regression and their principal differences.

4.A. Both Ridge and Lasso regressions are techniques designed to reduce model complexity and prevent over-fitting. Its main difference is that Ridge will never make any coefficient 0, whereas Lasso may do it if variables are not significant.

2.3 Q3. Non-Linear Regression

1.Q. Do you think a model of the family $f(x, \theta) = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2$ is a good choice for such task? Why?

1.A. No, because as it can be seen in the image, data can be better described with a quadratic equation, not a linear one.

2.Q. Do you think using a feed forward neural network with activation function $h(x) = 1 - 2 \cdot x \cdot e^{-2}$ would improve the results?

2.A. This function is still a linear one, so it will not change any results. Using a parabolic or sinusoidal one may improve feeding other nodes as activation function.

3.Q. Do you think it would be possible to achieve good performance with a linear model? How?

3.A. Yes, if the linear model supported quadratic equations. This data can be fitted into a quadratic formula, so if we can generate this model using a x^2 formula, it can produce good results.

4.Q. What is the purpose of the hidden activation function in a feed forward neural network?

4.A. The purpose of the hidden activation function in a feed forward neural network is to decide the output node in a given layer. It will transform the input as a summed weighted from the current node into an output value that will be sent to the next hidden layer.