Principles of Machine Learning

Methodology

Academic year 2021/2022

Queen Mary University of London

SOLUTIONS

EXERCISE ♯1 (SOL): The training MSE is used for the sole purpose of tuning a model whereas the test MSE gives us an estimation of the future performance of the model when deployed.

- When F is close to 1, most of the samples in the dataset are used for training, leaving a small number of samples for testing. In this scenario, we should expect randomness to have a lower impact on the value of the parameters of the final solution we find. The MSE surface during training will not change much with the actual samples used during training. Since we are using a limited number of samples for testing, the estimation of the deployment MSE will be very poor and will depend on the actual samples that have used. In other words, no matter whether the test MSE is high or low, we will not trust this number, as it has been obtained using a limited number of samples.
- When F is close to 0, most of the samples are used for testing. Since we are training our model on a small number of samples, we could expect the model to overfit to the training dataset. Hence, the training MSE will be close to 0. Since we are using most of our samples for testing, the test MSE value that we obtain will be very reliable. We will expect the test MSE to be very close to the real value, however, most likely this value will be very high, due to overfitting.

EXERCISE ♯2 (SOL):
If 30% of 1,000 samples (300 samples) are sufficient for estimate the deployment performance of a model, we would use 9,700 samples (97%) from the new dataset for training and 300 (3%) for validation. We would improve our trained models without compromising on the quality of our estimation of the deployment performance.

EXERCISE ♯3 (SOL):

- Your company will be deploying a machine learning model that someone else has built. Therefore, your company should define a target performance as well as comparison criteria. A test task would be needed before choosing a final solution to be deployed. It is important to note that your company's notion of deployment performance might be different from the developers', hence it is your company's responsibility to ensure that models work as intended. For instance, your company might want to ensure animals are detected too, whereas the developer might have focused solely on humans. A test dataset which is representative of the scenarios that self-driving cars will be expected to face can be created to test the performance of any candidate model.
- A training dataset can be provided with. This dataset needs to be different from the dataset that your company will use for testing and the latter dataset should never be shared outside your company. Otherwise, your company will not be able to establish whether a model works well during deployment or has been designed to work well on the test dataset.
- As a developer you could use the dataset to train many different families of models. However, you can only send one solution: hence you need to select the right family of models first and then train it. The available dataset can be used for validation purposes first, where we train and estimate the deployment performance of each family of models. Based on your estimation of the deployment performance, you would select one family of models. Then, you would train the model using the whole available dataset. The resulting solution would be sent to the car manufacturer, who would test it.

- The fourth model has the highest performance, namely 0.92. The second highest performance is 0.91, then 0.905, 0.9 and 0.89. All these figures are estimations of the deployment performance of a model, hence we should consider whether the differences are significant or might be due to random chance. In addition to machine learning performance, the car manufacturer might be considering other factors as well, e.g. model size, speed, etc. Before sending a final model for consideration, you would need to carefully evaluate all of these factors together.

EXERCISE ♯4 (SOL):

First, let's note that the way we extract samples from each dataset is random. However the true relationship between x and y is deterministic in the dataset 1, has both a deterministic and a random component in dataset 2, and in dataset 3 x and y are independent.

- Dataset 1: The training and validation MSE will be 0, as x and y have a deterministic relationship and the 4 polynomial models have sufficient flexibility to reproduce the underlying linear pattern. The polynomial coefficients will be 0 except for the coefficient of the linear term, which will be $w_1 = 3$. Increasing the number of samples will not have any impact.
- Dataset 2: The mean value of the Gaussian component is 2, which will manifest as a constant value in the relationship between x and y. As we increase the order of the polynomial, the risk of overfitting increases as well. The polynomials of order 3 and 4 have respectively 4 and 5 coefficients. Since we are using 4 samples for training, we should expect both to overfit: their training MSE will be 0, and the validation will be very high. For lower degrees, we should expect $w_0 = 2$ and $w_1 = 3$, remaining coefficients close to 0, training and validation MSE close to σ^2.
- Dataset 3: x and y are independent. Once again, polynomials of order 4 and 5 will overfit when we use 4 samples for training, training MSE will be 0 and validation MSE will be high. For polynomials of lower degree and when we increase the number of samples, we should expect $w_0 = 0.5$ (mean of the uniform distribution) and remaining coefficients close to 0. Training and validation errors should be close to the variance of the uniform distribution.

EXERCISE $\sharp$5 (SOL):

Training and test dataset need to be representative of the underlying population. Sometimes dataset reflect some structure that describing the sampling procedure, rather than the actual population. Randomisation avoids the confusion introduced by the sampling procedure itself. We would choose the second option.

EXERCISE $\sharp$6 (SOL):

First of all, let's note that our goal is to create a model that takes N values as an input and produces a single output. Hence, there are N predictors and one prediction.

- In this scenario, we are considering splitting the entire time series into two segments, the first one has K_1 prices (the oldest ones) and the second $K - K_1$ prices (the most recent ones). In other words, we are using old prices for training and recent prices for validation. If our time series is stationary, i.e. has the same behaviour in the past and in the present, we should expect this strategy to work. Otherwise, we might find a model that works well for past behaviour, but not so well for current behaviour.

- In this second scenario we are using old and recent prices for training and validation. Hence, we seem to be avoiding stationarity issues. However, we need to be careful not to use very similar segments for training and for validation. If segments overlap, we might end up validating our approach using segments that are very similar to the training ones, which would result in "data leakage" and a poor estimation of the true deployment performance. Hence, we need to extract non-overlapping segments.

- The value N reflects the memory of the system, i.e. how much predictive value a past price has on a current price. If N is small, we are assuming that only the most recent prices are useful to predict a future price. If it is large, we assume that distant prices are can be used to predict a future value. As we increase N, the segment size increases and the number of segments that we can extract decreases (the number of segments is K/N). Hence, since each segment is 1 sample (or item), increasing N results in a decrease in the number of training samples, hence our models will be trained on less data of higher dimensionality.