EECS 189 Introduction to Machine Learning
Fall 2020 DIS2

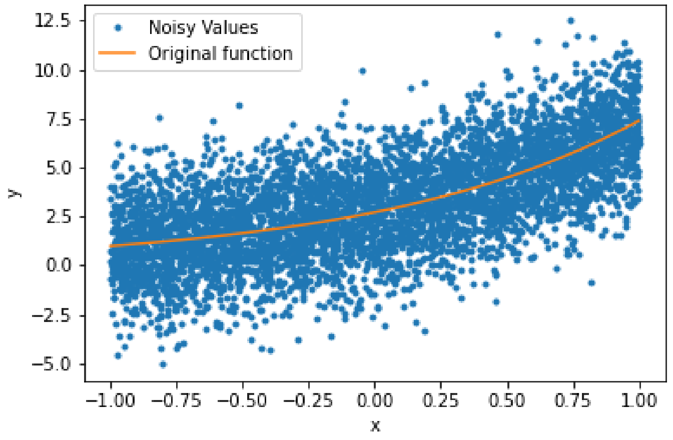
This discussion was released **Friday, September 11**.

1 Overview of test sets, validation, and cross-validation

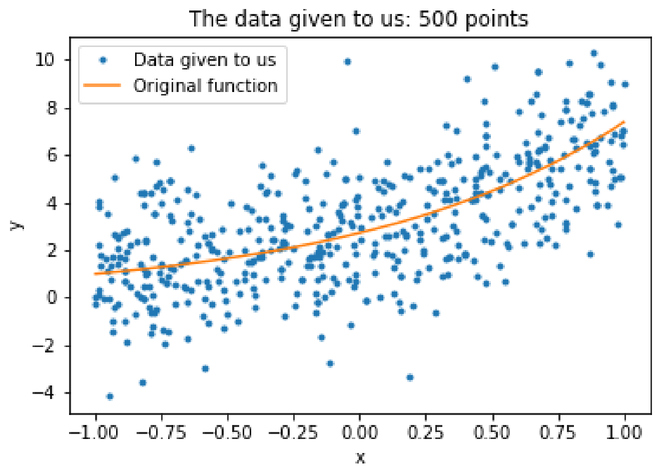
In this part, we discuss several issues having to do with test sets and the notions of validation and cross-validation. Open this notebook in datahub and discuss the questions it contains.

Solution:

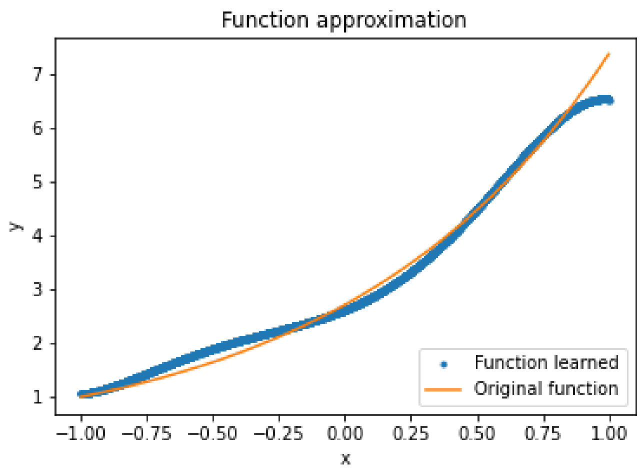
Test sets. The purpose of this notebook is to reflect about the concepts of test sets and validation. Upon executing the first cells, you will see the following plot:



The function we are trying to approximate is shown in orange, and the sampled data (with noise), in blue. Observe that the domain of the true function is the interval $[-1, 1]$. The next cell carries out a subsampling of the data. This is the data that we assume someone is giving to us in order to learn a function.



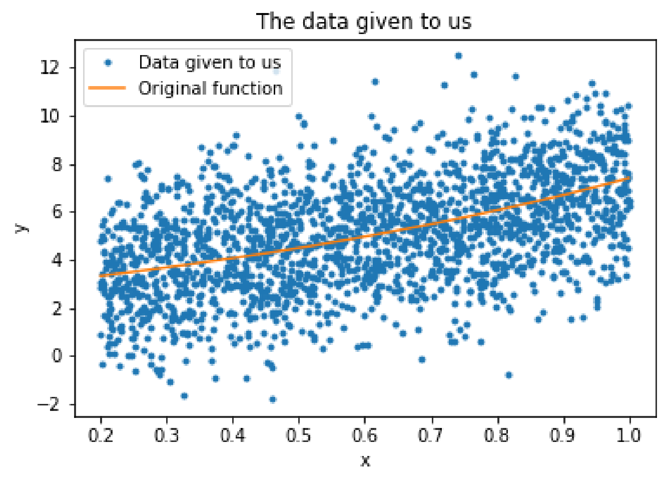
Immediately after seeing this plot, we have a cell with routines to split the data into a training and a test set, and to train a model. The model being trained in this cell is a fixed 5th-degree polynomial. Observe that there is a line of code to be added in order to carry out the correct splitting of the data into a training set and a test set. The line is `trainIndices = np.logical_not(testIndices)`.

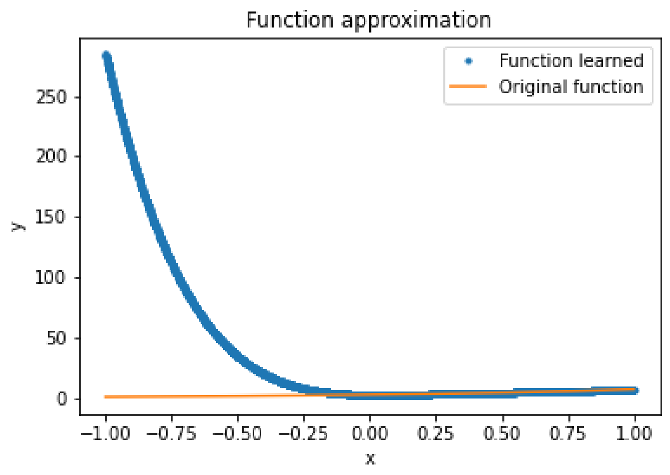


The plot we obtain immediately after the execution of this cell shows us the true function we are trying to learn, and the training and test errors. We also obtain this output from the execution of the cell:

```
Training error: 3.6511493057865483
Test error: 3.9256574783857046
True error: 0.03422772128092661
```

We observe that the test and training errors are closed to each other. Our data is noisy; this explains why in this case we see a test error bigger than the training error. We measure the true error with respect to the underlying true function, so this error is not directly affected by the noise in the training data. Immediately after this, we now sample the data with a bias: instead of random sampling, as happened before, the data given to us for training only contains values of $x > 0.2$. The following plots show the data extracted this time, and the result of fitting our data. Regarding the code used in the notebook, observe that the first time we fit the data, we built most routines by hand; in this case, we make more significant use of existing APIs.



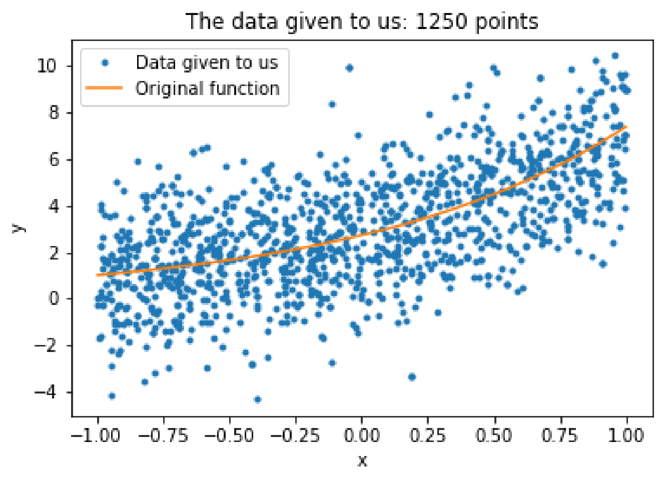


The last plot comes accompanied by this output:

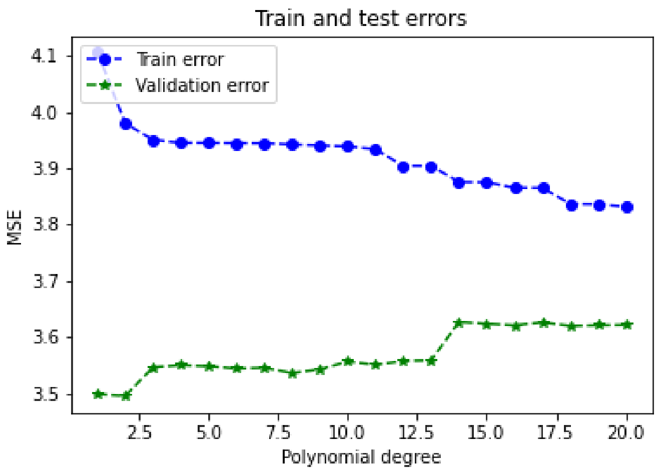
```
Training error: 3.9543414420385754
Test error: 3.588770415447887
True error: 5436.558416632976
```

We observe the training and test errors close to each other, as happened before, but now the true error is off-the-charts. The point here is that the data we used for training does not represent the entire domain of our x values. In this case, we are trying to make predictions outside the interval for which we had x values.

Validation. The next plots deal with the concept of validation. The first plot shows the data we will use during training. We use the function we introduced during our discussion of test sets. The data is randomly sampled from the entire interval $[-1, 1]$



The code executed immediately after this plot does the following: it breaks the data *given to us* into training set, validation set, and test set. We train polynomials of varying degrees to our data, and we plot the training and validation errors we obtain for each:



This plot tells us that we should pick a degree equal to 2 because this yields the lowest validation error. Using the code given in the notebook, we can compute the test error for a polynomial of degree 2. We obtain 3.19. Observe that validation is used as a methodological tool to pick a hyperparameter. Its drawback is that it prevents us to use all our training data in our optimization routines that fit our models.

Cross-validation. If we reserve k points for our validation set, we have this tradeoff: a low k means our validation set is too small and can't yield a reliable estimate of the true error; if we set k

too high, validation yields an excellent estimate of the true error, but our models will be trained on little data. Leave-one-out cross-validation is an example of cross-validation. This technique allows us to use all data during training, but it requires us to train $\sim nm$ models, where n is the size of the training set, and m is the number of evaluations of the hyperparameters. This is unrealistic, but the idea is intriguing: we have a method that allows to train on the entire dataset while getting a handle on the generalization error. One way to address this disadvantage of leave-one-out is K-fold cross-validation. You will learn about this technique soon!

The following is Problem 4.d from HW2. This is a standalone problem, i.e., it does not depend on results from 4.a–4.c.

2 Outlier Removal via OMP (Part 2)

(a) From the law of large numbers, we have seen that with a large number of samples, the sample mean converges to the population mean or expected value. More rigorously, the weak law of large numbers states the following: For any positive number ε,

$$\lim_{n\to\infty} \mathbb{P}\left(\left|\overline{X}_n - \mu\right| > \varepsilon\right) = 0$$

where μ is the expectation, and $\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$ is the sample mean. Here, we would like to make a similar statement for sample median and population median. Given a sequence of n random variables i.i.d. drawn from the same distribution, $\{X_1, X_2, ..., X_n\}$, let's denote the population median as $med(X)$ and the sample median as $\tilde{X}_n$. We want to make no other assumption on the distribution of X_i's. The goal is to give a proof of the following statement:

$$\lim_{n\to\infty} \mathbb{P}\left(\left|\tilde{X}_n - med(X)\right| > \epsilon\right) = 0$$

But to make the proof easier to follow and to understand things in terms of their natural dependencies, we will modify the above statement to involve *quantiles* of X. For every $\varepsilon > 0$ for which the $(\frac{1}{2} - \varepsilon)$ quantile is different from the median and the $(\frac{1}{2} + \varepsilon)$ quantile is also different from the median, we have:

$$\lim_{n\to\infty} \mathbb{P}\left(\tilde{X}_n < (1/2-\varepsilon)\text{-quantile} \;\text{ or }\; \tilde{X}_n > (1/2+\varepsilon)\text{-quantile}\right) = 0.$$

Here, (for simplicity) a p-quantile of a random variable is a value x for which the CDF $\mathbb{P}(X \le x) = p$. [To be precise, a p-quantile is an x for which $\mathbb{P}(X < x) \le p$ and $\mathbb{P}(X \le x) \ge p$. This allows the distribution of X to have atoms in it and for quantiles to still be defined in a reasonable manner.] Notice that by choosing an appropriate value of ε, we can recover the desired ϵ, and hence, the two statements are equivalent. *(First hint: Consider a Bernoulli random variable: $Y_i = \mathbb{1}\{X_i > (1/2+\varepsilon)\text{-quantile}\}$) (Second hint: Think about a relevant Chernoff bound and use it. You don't have to use a Chernoff bound to prove it, but it helps in understanding the speed of this convergence.)* **Solution:** To be released as part of the solutions to HW2.

Contributors:

- Anant Sahai
- Chawin Sitawarin
- Inigo Incer