Labs
**Machine Learning Course**
Fall 2022

**EPFL**
School of Computer and Communication Sciences
**Nicolas Flammarion & Martin Jaggi**
www.epfl.ch/labs/mlo/machine-learning-cs-433

# Problem Set 4, Oct 13, 2022 (Cross-Validation and Bias-Variance Decomposition)

**Goals.** The goal of this exercise is to

- Implement $4$-fold cross-validation.
- Understand the bias-variance decomposition.

**Setup, data and sample code.** Obtain the folder `labs/ex04` of the course github repository

github.com/epfml/ML_course

## 1 Cross-validation

This exercise is partly based on the materials from last week (`labs/ex03`). If you don't have it already, please finish the ex03 first. You might directly reuse/copy some of the functions you implemented yourself during previous exercises, e.g., `ridge_regression()`, `least_squares()` and `build_poly()`.

**Exercise 1:**

Implement $4$-fold cross-validation.

- Copy your code from last week, and fill in the corresponding templates, i.e., `ridge_regression()` to `ridge_regression.py`, `build_poly()` to `build_polynomial.py`, and `least_squares()` to `least_squares.py`.

  In this exercise, please fill in the notebook functions `cross_validation()` and `cross_validation_demo()`, and perform $4$-fold cross-validation for polynomial degree 7. Plot the train and test RMSE as a function of $\lambda$. The resulting figure should look like Figure 1.
- How will you use $4$-fold cross-validation to select the best model among various degrees, say from 2 to 10? Write code to do it in `best_degree_selection()`.

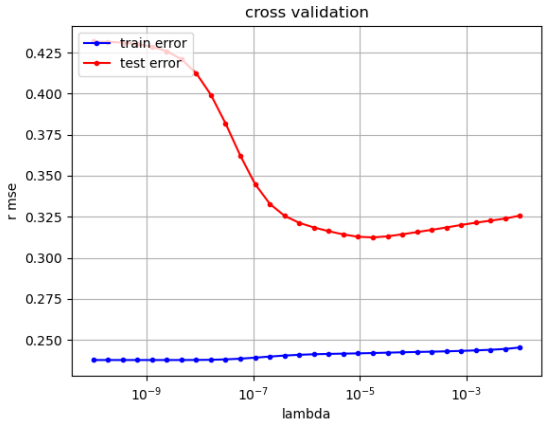


Figure 1: Effect of $\lambda$ on training and test errors, calculated using $4$-fold cross-validation

## 2 Visualizing the Bias-Variance Decomposition

Last lecture we introduced model selection, and we saw that the model complexity is crucial to the performance. In this problem, we will further investigate the effect of *model complexity* with the concept of *bias-variance decomposition*.

We will implement the figures seen in class representing the tradeoff and also seen in Figure 2 : for a big polynomial degree, the bias is small but the variance is large. The opposite is true for a small polynomial degree (however notice that the variance is still quite important). Choosing an intermediate degree leads to consistent predictions which are close to the true function we want to learn (optimal bias / variance tradeoff).

**Exercise 2:**

Visualizing the bias-variance trade-off.

- Complete the notebook function `bias_variance_one_seed()`: for $15$ random datapoints, it finds the optimal fit (using the least square formula, with no regularisation $\lambda$) for a polynomial expansion of degree $1$, $3$ and $6$.
- you can play around by changing the seed, the number of datapoints, the degree of the polynomial expansion etc.
- Now complete the notebook function `bias_variance_demo()` which performs many times the previous experiment but with a new random training set each time. You should obtain something similar to Figure 2.
- Comment the figures by explaining how the bias / variance tradeoff is shown in these plots.
- You can play around by changing the function you want to learn, the variance of the gaussian noise $\sigma^2$, the degree of the polynomial expansion etc.
- **BONUS:** you can do similar figures but now you fix the degree of the polynomial expansion and add some regularisation $\lambda$. You will observe a similar bias / variance tradeoff when changing the magnitude of the regularisation.

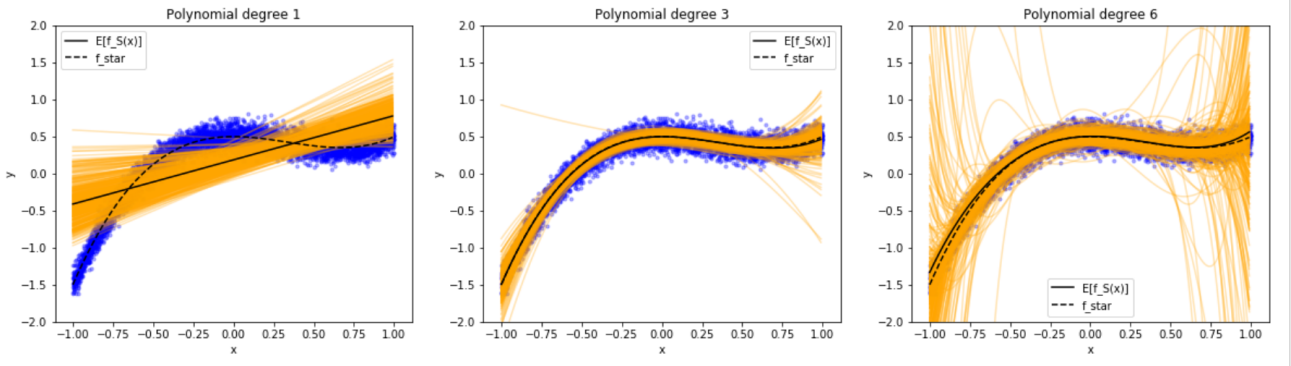


Figure 2: Visualizing the Bias-Variance Trade-off.