PolyCurveFitting

October 20, 2026

1 Polynomial Curve Fitting

In this assignment, you are asked to run an experimental evaluation of polynomial curve fitting on an artificial dataset with and without L2 regularization.

1.1 Write Your Name Here:

2 Submission instructions

1. Click the Save button at the top of the Jupyter Notebook.
2. Please make sure to have entered your name above.
3. Select Cell -> All Output -> Clear. This will clear all the outputs from all cells (but will keep the content of ll cells).
4. Select Cell -> Run All. This will run all the cells in order, and will take several minutes.
5. Once you've rerun everything, select File -> Download as -> PDF via LaTeX and download a PDF version showing the code and the output of all cells, and save it in the same folder that contains the notebook file.
6. Look at the PDF file and make sure all your solutions are there, displayed correctly.
7. Submit **both** your PDF and the notebook file .ipynb on Canvas.
8. Make sure your your Canvas submission contains the correct files by downloading it after posting it on Canvas.

2.1 Theory question on linear regression (10p)

Consider the following regression problem, where the input contains two features:

	x_1	x_2	y
$\mathbf{x}^{(1)}$	30.2	2.1	63.3
$\mathbf{x}^{(2)}$	65.4	3.5	229.1
$\mathbf{x}^{(3)}$	46.2	0.5	22.9
$\mathbf{x}^{(4)}$	27.6	4.1	109.1
$\mathbf{x}^{(5)}$	25.3	4.0	y_5

1. What do you think would be a good estimate for the label y_5 of the test example? Explain.

- *Hint: The features and the labels are noisy measurements of something …*

2. If a multiple linear regression were trained on such data, would it be able to make good predictions? If yes, explain why. If not, explain how you would transform the input data so that a multiple linear regression model trained on it made good predictions. Explain.

2.1.1 Your answers go here

1. Estimate of y_5:
2. Multiple linear regression setting:

2.2 Read dataset, split into training, test, and development

The file `../data/polyfit/dataset.txt` contains two columns of numbers:

- The first column contains 30 values x_n uniformly spaced in the unit interval $[0, 1]$.
- The second column contains the corresponding labels y_n that were generated according to $y(x) = sin(2\pi x) + x(x+1)/4 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.005)$ is a zero mean Gaussian noise with variance 0.005.

[ ]:
```
# Import necessary Python modules.
import numpy as np
from matplotlib import pyplot as plt
```

2.2.1 Read the dataset (5p)

Read the dataset from the text file `../data/polyfit/dataset.txt`.

[ ]:
```
# Read data samples x and labels y from text file.
def read_data(file_name):
    data = np.loadtxt(file_name)

    # YOUR CODE HERE
    x = []
    y = []

    return x, y

x, y = read_data('../data/polyfit/dataset.txt')
```

2.2.2 Split dataset into trainining, test, and development (5p)

Create training examples (x_n, y_n) for $n \in \{0, 3, 6, ..., 27\}$, store them in `x_train` and `y_train`.

Create test examples (x_n, y_n) for $n \in \{1, 4, 7, ..., 28\}$, store them in `x_test` and `y_test`.

Create development examples (x_n, y_n) for $n \in \{2, 5, 8, ..., 29\}$, store them in `x_devel` and `y_devel`.

[ ]:
```
x_train = # YOUR CODE HERE
y_train = # YOUR CODE HERE

x_test = # YOUR CODE HERE
```

```
y_test = # YOUR CODE HERE

x_devel = # YOUR CODE HERE
y_devel = # YOUR CODE HERE
```

2.2.3 Plot the dataset (5p + 5p)

Plot the dataset, using the horizontal axis for x and vertical axis for y. Use small green triangles for training, small blue circles for testing, and small red squares for development samples.

- You can use Matplotlib functions such as plt.plot() or plt.scatter(). Here is a short tutorial.
- For bonus points, on the same graph also plot the data generating function $y(x) = sin(2\pi x) + x(x+1)/4$ using a black color.

[1]:
```
# YOUR CODE HERE
```

2.3 Polynomial curve fitting, no regularization (30p)

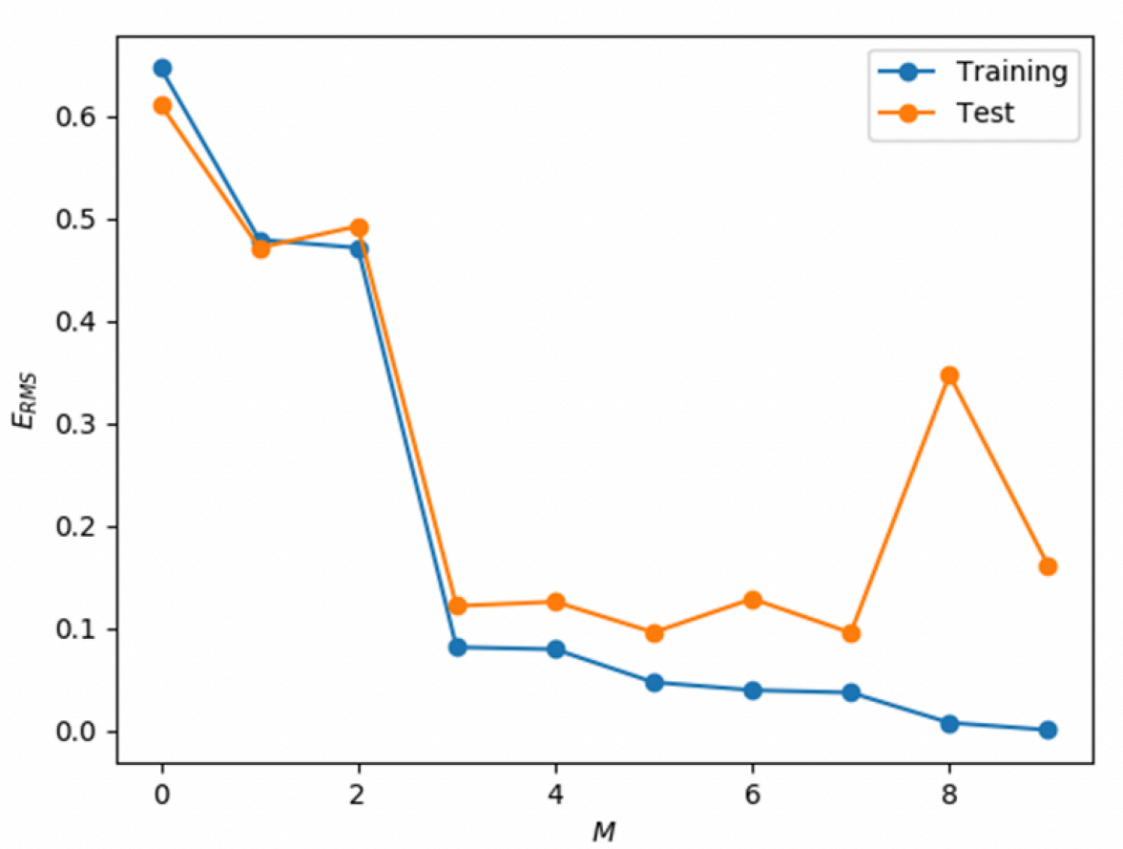
Use the training data to infer the parameters **w** for all values of $M \in [0, 9]$. For each order M , compute the RMSE separately for the training and test data, and plot all the values on the same graph, as shown in class (slide 19).

- To find the parameters, you can solve the system of linear equations for polynomial curve fitting (slide 11), or the vectorized normal equations (slide 12). This can be done in oen line of NumPy code.
- Feel free to reuse code from previous assignments.

Your plot should look like the one below.

[4]:
```
from IPython.display import Image
Image('rmse-vs-m.png', width = 500)
```

[4]:



[3]:
```
# YOUR CODE HERE
```

2.4 Polynomial curve fitting, with L2 regularization (30p)

Fixing $M = 9$, use the training data to infer the parameters $\mathbf{w}$, one parameter vector for each value of $\ln \lambda \in [-50, 0]$ in steps of 5.

For each parameter vector that you get, compute the RMSE separately for the training and validation data, and plot all the values on the same graph, as shown in class (slide 26, where you use validation instead of test).

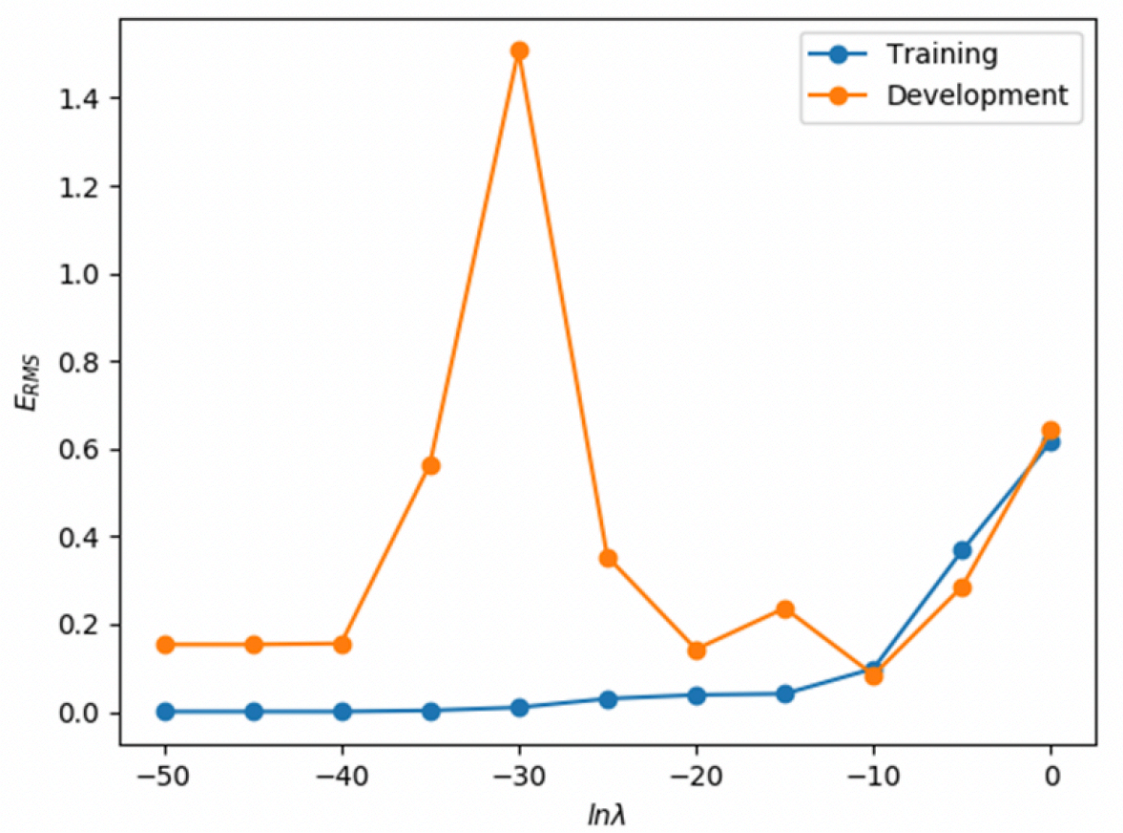
Select the regularization parameter λ that leads to the parameter vector that obtains the lowest RMSE on the validation data, and use it to evaluate the model on the test data. Report and compare the test RMSE with the one obtained without regularization.

- Hint: For each value of $\lambda \in \{e^{-50}, e^{-45}, ..., e^{-5}, e^{0}\}$, finding the parameters $\mathbf{w}$ is done by using the vectorized normal equations for ridge regression (slide 31).
- For bonus points, use the numeral equations updated such that w_0 is not part of the L2 regularization term (as shown in class).

Your plot should look like the one below.

[5]:
```
from IPython.display import Image
Image('rmse-vs-ln.png', width = 500)
```

[5]:



[2]:
```
# YOUR CODE HERE
```

Report and compare here the test RMSE with the one obtained without regularization.

- RMSE on test data, no regularization:
- RMSE on test data, with L2 regularization:

Analysis:

2.5 Bonus points

- (20p) Train and evaluate the polynomial curve fitting models above using `sklearn`. For ridge regression, add the validation data tothe training data and use the RidgeCV function to tune the hyper-parameter λ (use 10 folds). Report and compare the test RMSE with vs. without regularization.
- Anything extra that is non-trivial.

[ ]:
```

```