Exercises 5.7 **Polynomial basis feature regression for scalar valued input**

In this exercise you will explore how various degree D polynomial basis features fit the sinusoidal dataset shown in Fig. 5.12. You will need the wrapper *poly_regression_hw* and the data located in *noisy_sin_samples.csv*.

a) Use the description of polynomial basis features given in Example 5.1 to transform the input using a general degree D polynomial. Write this feature transformation in the module

$$\mathbf{F} = \text{poly\_features}\,(\mathbf{x},\, D) \tag{5.41}$$

located in the wrapper. Here $\mathbf{x}$ is the input data, D the degree of the polynomial features, and $\mathbf{F}$ the corresponding degree D feature transformation of the input (note your code should be able to transform the input to any degree D desired).

b) With your module complete you may run the wrapper. Two figures will be generated: the first shows the data along with various degree D polynomial fits, and the second shows the mean squared error (MSE) of each fit to the dataset. Discuss the results shown in these two figures. In particular describe the relationship between a model's MSE and how well it seems to represent the phenomenon generating the data as D increases over the range shown.

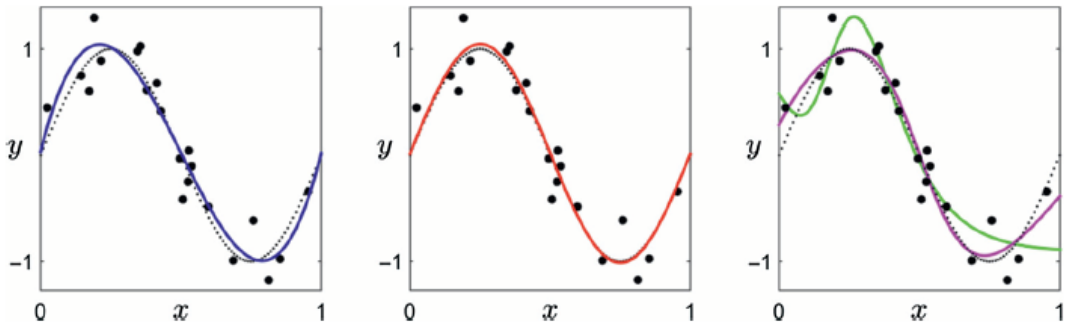

Fig. 5.12

Example 5.1 **Regression with fixed bases of features**

To perform regression using a fixed basis of features (e.g., polynomials or Fourier) it is natural to choose a degree D and transform the input data using the associated basis functions. For example, employing a degree D polynomial or Fourier basis for a scalar input, we transform each input x_p to form an associated feature vector $\mathbf{f}_p = \begin{bmatrix} x_p & x_p^2 & \cdots & x_p^D \end{bmatrix}^T$ or $\mathbf{f}_p = \begin{bmatrix} \cos\left(2\pi x_p\right) & \sin\left(2\pi x_p\right) & \cdots & \cos\left(2\pi D x_p\right) & \sin\left(2\pi D x_p\right) \end{bmatrix}^T$ respectively. For higher dimensions of input N fixed basis features can be similarly used; however, the sheer number of elements involved (the length of each $\mathbf{f}_p$)

explodes[11] for even moderate values of N and D (as we will see in Section 7.1, this issue can be ameliorated via the notion of a "kernal," however, this introduces a serious numerical optimization problem as the size of the data-set grows).

In any case, once feature vectors $\mathbf{f}_p$ have been constructed using the data we can then determine proper weights b and $\mathbf{w}$ by minimizing the Least Squares cost function as

$$\underset{b,\mathbf{w}}{\text{minimize}} \sum_{p=1}^{P} \left(b + \mathbf{f}_p^T\mathbf{w} - y_p\right)^2. \tag{5.18}$$

Using the compact notation $\tilde{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$ and $\tilde{\mathbf{f}}_p = \begin{bmatrix} 1 \\ \mathbf{f}_p \end{bmatrix}$ for each p we may rewrite the cost as $g\left(\tilde{\mathbf{w}}\right) = \sum_{p=1}^{P} \left(\tilde{\mathbf{f}}_p^T\tilde{\mathbf{w}} - y_p\right)^2$, and checking the first order condition then gives the linear system of equations

$$\left(\sum_{p=1}^{P} \tilde{\mathbf{f}}_p\tilde{\mathbf{f}}_p^T\right)\tilde{\mathbf{w}} = \sum_{p=1}^{P} \tilde{\mathbf{f}}_p y_p, \tag{5.19}$$

that when solved recovers an optimal set of parameters $\tilde{\mathbf{w}}$.

Exercises 5.10 Four guys and four error plots

Eric, Stanley, Kyle, and Kenneth used hold out cross-validation to find the best degree polynomial fit to their respective datasets. Based on the error plots they have made (Fig. 5.21), what advice would you give to each of them as to what their next step should be.

Fig. 5.21 Hold out cross-validation error plots for four different datasets.

Exercises 5.11 **Practice the hold out method**

In this exercise you will perform hold out cross-validation on the dataset shown in Fig. 5.16 as described in Example 5.5 of the text. Start by randomly splitting the dataset, located in *wavy_data.csv*, into $k = 3$ equal sized folds (keeping 2 folds as training, and 1 fold as testing data). Use the Fourier basis features M in the range $M = 2, 4, 6, \ldots, 16$ (or likewise D in the range $D = 1, 2, \ldots, 8$) and produce a graph showing the training and testing error for each D like the one shown in Fig. 5.16, as well as the best (i.e., the lowest test error) model fit to the data.

Note: your results may appear slightly different than those of the figure given that you will likely use a different random partition of the data. Note: you may find it very useful here to re-use code from previous exercises e.g., functions that compute Fourier features, plot curves, etc.

Example 5.5 **Hold out for regression using Fourier features**

To solidify these details, in Fig. 5.16 we show an example of applying hold out cross-validation using a dataset of $P = 30$ points generated via the function $y(x)$ shown in Fig. 5.3. To perform hold out cross-validation on this dataset we randomly partition it into $k = 3$ equal-sized (ten points each) non-overlapping subsets, using two partitions together as the training set and the final part as testing set, as illustrated in the left panel of Fig. 5.16. The points in this panel are colored blue and yellow indicating that they belong to the training and testing sets respectively. We then train our model on the training set (blue points) by solving several instances of the Least Squares problem in (5.18). In particular we use a range of even values for M Fourier features $M = 2, 4, 6, \ldots, 16$ (since Fourier elements naturally come in pairs of two as shown in Equation (5.7)) which corresponds to the range of degrees $D = 1, 2, 3, \ldots, 8$ (note that for clarity panels in the figure are indexed by D).

Based on the models learned for each value of M (see the middle set of eight panels of the figure) we plot training and testing errors (in the panel second from the right), measuring how well each model fits the training and testing data respectively, over the entire range of values. Note that unlike the testing error, the training error always decreases as we increase M (which occurs more generally regardless of the dataset/feature basis used). The model that provides the smallest testing error ($M^\star = 10$ or equivalently $D^\star = 5$) is then trained again on the entire dataset, giving the final regression model shown in red in the rightmost panel of Fig. 5.16.

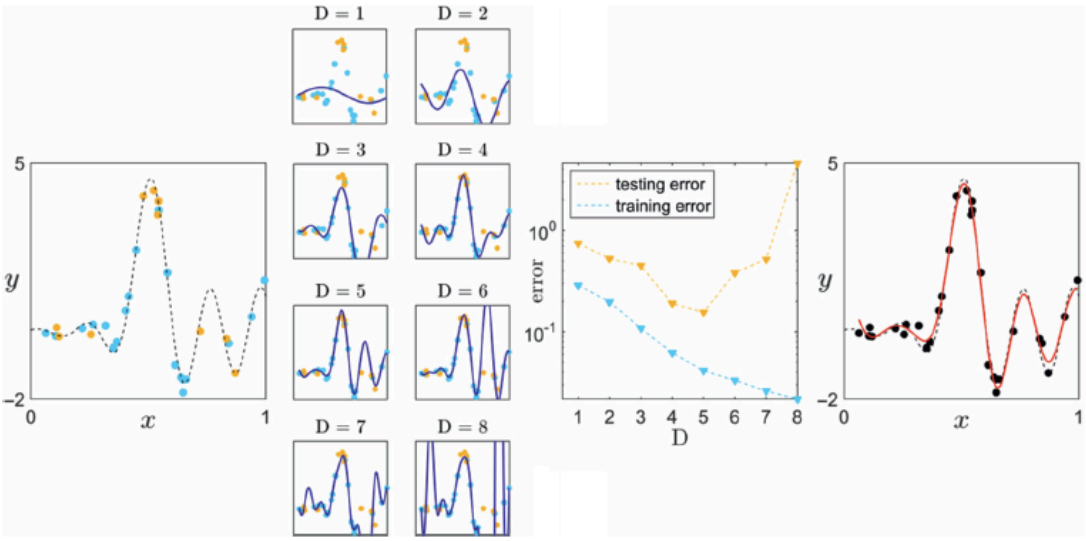

Fig. 5.16

Exercises 5.12 **Code up k-fold cross-validation**

In this exercise you will perform k-fold cross-validation on the dataset shown in Fig. 5.19 as described in Example 5.7 of the text. Start by randomly splitting the dataset of $P = 6$ points, located in *galileo_ramp_data.csv*, into $k = 6$ equal sized folds (keeping 5 folds as training, and 1 fold as testing data during each round of cross-validation). The value of $k = P$ has been chosen in this instance due to the small size of the dataset (this is sometimes called "leave one out" cross-validation since each training set consists of all but one point from the original dataset).

Use the polynomial basis features and M in the range $M = 1, 2, \ldots, 6$ and produce a graph showing the average training and testing error for each M, as well as the best (i.e., the lowest average test error) model fit to the data.

Note: you may find it very useful here to re-use code from previous exercises, e.g., functions that split a dataset into k random parts, that compute training/testing errors, polynomial features, plot curves, etc.

Example 5.7 **Leave-one-out cross-validation for Galileo's ramp data**

In Fig. 5.19 we show how using $k = P$ fold cross-validation (since we have only $P = 6$ data points, intuition suggests, see Section 5.3.2, that we use a large value for k), sometimes referred to as *leave-one-out cross-validation*, allows us to recover precisely the quadratic fit Galileo made by eye. Note that by choosing $k = P$ this means that every data point will take a turn being the testing set. Here we search over the polynomial basis features of degree $M = 1 \ldots 6$. While not all of the hold out models over the six folds fit the data well, the average k-fold result is indeed the $M^\star = 2$ quadratic polynomial fit originally proposed by Galileo!

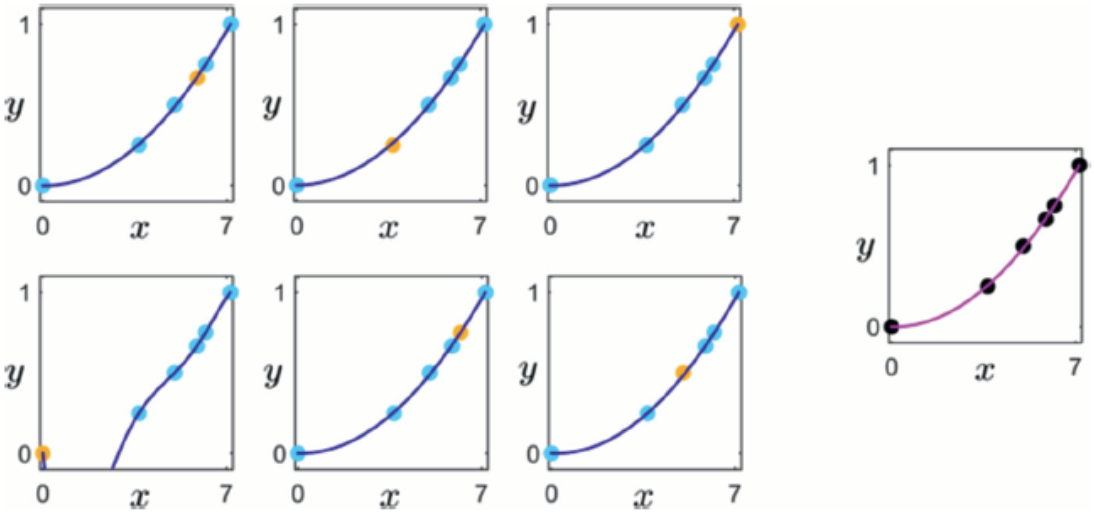

Fig. 5.19