https://doi.org/10.17163/abyaups.146.2

PART II

REGRESSION

You will see here, at last, a real Machine Learning algorithm! Specifically, *linear regression*, a powerful algorithm that can solve even non-linear regression problems, in spite of its humble name. In Chapter 5, you will learn not only how to have MATLAB executing this algorithm for you, but also the tricks to perform non-linear regression with it. In addition, in Chapter 6 we will take advantage of the beauty of the curves that represent linear-regression solutions in the one-dimensional case, to visually show you two awful problems we deal with in Machine Learning: *overfitting* and *underfitting*.

CHAPTER 5

LINEAR REGRESSION

5.1 Theoretical Briefing

Linear regression model. Let $\mathbf{X}$ and $\mathbf{y}$ be the data matrix and the dependent variable for a regression problem, respectively. The linear regression model is a mathematical model given by:

$$y_1 = w^{(0)} + w^{(1)}x^{(1)} + w^{(2)}x^{(2)} + \cdots + w^{(0)}dx^{(d)}$$

for each of the instances of this training set. This is, each of the following equations should hold true:

$$\begin{aligned}
&1.\ y_1 = w^{(0)} + w^{(1)}x_1^{(1)} + w^{(2)}x_1^{(2)} + \cdots + w^{(d)}x_1^{(d)} \\
&2.\ y_2 = w^{(0)} + w^{(1)}x_2^{(1)} + w^{(2)}x_2^{(2)} + \cdots + w^{(d)}x_2^{(d)} \\
&\quad\vdots \\
&m.\ y_m = w^{(0)} + w^{(1)}x_m^{(1)} + w^{(2)}x_m^{(2)} + \cdots + w^{(d)}x_m^{(d)}
\end{aligned}$$

This system can be written as a matrix equation:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(d)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(d)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m^{(1)} & x_m^{(2)} & \cdots & x_m^{(d)} \end{bmatrix} \begin{bmatrix} w^{(0)} \\ w^{(1)} \\ \vdots \\ w^{(d)} \end{bmatrix}$$

Or, in a compact way:

$$\mathbf{y} = \mathbf{X}_{aum} \cdot \mathbf{w} \tag{5.1}$$

Linear regression. The last system of equations is an overdetermined system (it has more equations than unknowns) and it is impossible to find a solution which satisfies all the equations at once. Instead of that, however, we can find

a vector $\mathbf{w}$ which *approximately* satisfies all the equations at once. The mathematical procedure used to find such a solution is called "linear regression".

Method of least squares. One possibility for doing linear regression is the so-called "method of least squares", which is meant to minimise the mean squared error (see Equation 4.1). We will not explain the method of least squares here, we will only tell you how to have MATLAB applying it for you. Just type:

```
>> w = Xaum\y;
```

A mnemonic way to recall this expression is to write Equation 5.1 and then solve for $\mathbf{w}$. Multiplying both sides by $\mathbf{X}_{aum}^{-1}$:

$$\mathbf{X}_{aum}^{-1} \cdot \mathbf{y} = \mathbf{X}_{aum}^{-1} \cdot \mathbf{X}_{aum} \cdot \mathbf{w}$$

The inverse of a matrix multiplied by the matrix itself is the identity matrix, so that:

$$\mathbf{X}_{aum}^{-1} \cdot \mathbf{y} = \mathbf{I} \cdot \mathbf{w}$$

This results in:

$$\mathbf{X}_{aum}^{-1} \cdot \mathbf{y} = \mathbf{w}$$

This last expression, $\mathbf{w} = \mathbf{X}_{aum}^{-1} \cdot \mathbf{y}$, should remind you of the MATLAB code written above.

Polynomial regression. Polynomial regression involves higher-degree models. For example, suppose you have only one-dimensional vectors in your data matrix X:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_m^{(1)} \end{bmatrix}$$

To simplify, we can write this as:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Then, you can perform a second-degree polynomial regression (quadratic model) for this data, by creating a matrix

$$\mathbf{X}_{aum} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 \end{bmatrix}$$

and using the same code

```
>> w = Xaum\ y;
```

with it.

To perform a third-degree polynomial regression (cubic model), create the matrix:

$$\mathbf{X}_{aum} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 & x_m^3 \end{bmatrix}$$

and so on.

5.2 Exercises

Solve the following exercises. You can use a calculator. Do not use MATLAB here

Let $\mathbf{X}$ be the matrix

$$\mathbf{X} = \begin{bmatrix} -2 \\ 0 \\ 3 \\ 5 \\ 6 \end{bmatrix}$$

and let $\mathbf{y}$ be the vector

$$\mathbf{y} = \begin{bmatrix} 5 & 1 & -1 & -3 & -6 \end{bmatrix}^T$$

be the data for a regression problem.

1. Suppose you want to fit a linear model using linear regression (least-squares solution) to these data. Write down the corresponding system of equations (5 equations).
2. Write this system as a matrix equation.
3. The least-squares solution for this system gives as a result the following weight vector:
$$\mathbf{w} = \begin{bmatrix} 2.0885 \\ -1.2035 \end{bmatrix}$$
Compute the $\hat{\mathbf{y}}$ vector corresponding to this $\mathbf{w}$.
4. Compute the mean squared error for this prediction.

5. Suppose now that you want to fit a cubic model using linear regression (least-squares solution) to these data. Write down the corresponding system of equations (5 equations).

6. Write this system as a matrix equation.

7. The least-squares solution for this system gives as a result the following weight vector:

$$\mathbf{w} = \begin{bmatrix} 0.9112 \\ -1.1054 \\ 0.3563 \\ -0.0602 \end{bmatrix}$$

Compute the $\hat{\mathbf{y}}$ vector corresponding to this $\mathbf{w}$.

8. Compute the mean squared error for this prediction.

5.3 Practical in MATLAB

General Objective:

To make the student capable of computing and interpreting the linear regression solution for a regression problem.

Specific Objectives:

- The student should be able to compute the simple linear regression solution for a problem using MATLAB.
- The student should be able to plot the linear regression solution in order to visually interpret the result.
- The student should be able to compute the simple linear regression solution with multiple features for a problem using MATLAB.

Materials:

Computer, MATLAB.

Procedure:

Solve the following exercises in MATLAB

1. Create a script and save it following the format `Pr5_nameLastname.m`. You will write your code for the next numerals in this script.

2. In folder *Data Sets* you will find a file named "`cars.xlsx`". This file contain the real information the author of this book collected to buy a second hand car in 2016. This information includes (as you can see by opening

the file in Excel): distance travelled, age, engine capacity and price. Save `cars.xlsx` to the Current Folder and load it using `xlsread`. Store the data in a variable you will call "`carData`".

3. Extract the first column of this `carData` matrix (the distance travelled by the cars) and call this vector "`X_km`".

4. Extract the last column (the price of the cars) and call this vector "`ycars`". This will be considered as the dependent variable in this regression problem.

5. Write a function "`myLinearRegression(X,y)`" which takes a matrix `X` of size $m \times d$ (where m is the number of instances and d is the dimensionality of the vectors) and a vector `y` of labels (values of the dependent variable). This function should return a vector `yhat` of the predictions after performing linear regression. More specifically, the function must perform a simple linear regression, using the "\" MATLAB operator. (Challenge: Do NOT use `for` loops when writing this function.)

6. Use your function `myLinearRegression` to obtain a vector `yhat_km` corresponding to the values in `X_km`.

7. Use your function `plotRegression` (written in Practical 1.4) to plot the values in `ycars` and `yhat_km` against those in `X_km`. Add suitable labels to the axes.

8. By looking at the plot obtained in the last numeral, find the point which is below the green line and is farthest from it. This corresponds, roughly, to the car that is the "best buy". What is the price and the distance travelled by this car?

9. Load into the workspace the matrices contained in the file `matricesPr5.mat`. This file can be found in the folder *Data Sets*.

10. Use your function `myLinearRegression` to obtain a vector `yhat_1`, the linear regression solution corresponding to matrix `X` and vector `y` loaded in numeral 9. Use `plotRegression` to plot the values in `y` and `yhat_1` against those in `X`.

11. Use your function `computeMSE` (written in Practical 1.4) to compute the mean squared error of the predictions in `yhat_1`.

12. Use your function `myLinearRegression` to obtain a vector `yhat_2`, the predictions given by the quadratic model fit to the data. Plot this solution.

13. Compute the mean squared error of the predictions in `yhat_2`.
14. Use your function `myLinearRegression` to obtain a vector `yhat_3`, the predictions given by the `cubic` model fit to the data. Plot this solution.
15. Compute the mean squared error of the predictions in `yhat_3`.
16. Use your function `myLinearRegression` to obtain a vector `yhat_2`, the predictions given by the 4th-grade model fit to the data. Plot this solution.
17. Compute the mean squared error of the predictions in `yhat_4`.

5.4 Answers to selected exercises

Exercises

1. $$\begin{cases} w^{(1)} - 2w^{(2)} &= 5 \\ w^{(1)} &= 1 \\ w^{(1)} + 3w^{(2)} &= -1 \\ w^{(1)} + 5w^{(2)} &= -3 \\ w^{(1)} + 6w^{(2)} &= -6 \end{cases}$$

2. $$\begin{bmatrix} 1 & -2 \\ 1 & 0 \\ 1 & 3 \\ 1 & 5 \\ 1 & 6 \end{bmatrix} \times \begin{bmatrix} w^{(1)} \\ w^{(2)} \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ -1 \\ -3 \\ -6 \end{bmatrix}$$

3. $\hat{\mathbf{y}} = \begin{bmatrix} 4.4956 \\ 2.0885 \\ -1.5221 \\ -3.9292 \\ -5.1327 \end{bmatrix}$

4. 0.6655

5. $$\begin{cases} w^{(1)} - 2w^{(2)} + 4w^{(3)} - 8w^{(4)} &= 5 \\ w^{(1)} &= 1 \\ w^{(1)} + 3w^{(2)} + 9w^{(3)} + 27w^{(4)} &= -1 \\ w^{(1)} + 5w^{(2)} + 25w^{(3)} + 125w^{(4)} &= -3 \\ w^{(1)} + 6w^{()} + 36w^{(3)} + 216w^{(4)} &= -6 \end{cases}$$

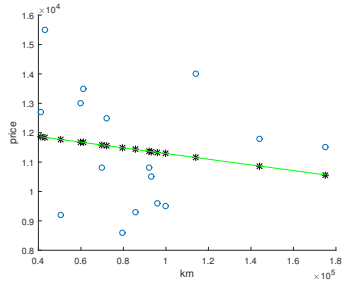
6. $$\begin{bmatrix} 1 & -2 & 4 & -8 \\ 1 & 0 & 0 & 0 \\ 1 & 3 & 9 & 27 \\ 1 & 5 & 25 & 125 \\ 1 & 6 & 36 & 216 \end{bmatrix} \times \begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ -1 \\ -3 \\ -6 \end{bmatrix}$$

7. $\hat{\mathbf{y}} = \begin{bmatrix} 5.0285 \\ 0.9112 \\ -0.8225 \\ -3.2282 \\ -5.8890 \end{bmatrix}$

8. 0.0209

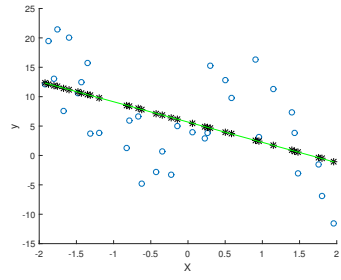
Practical

7.



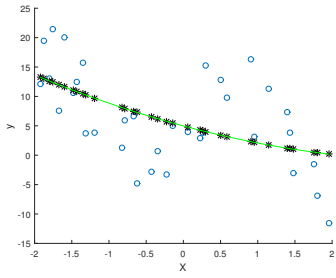
8. Price: 8600 USD; Distance travelled: 79600 km

10.



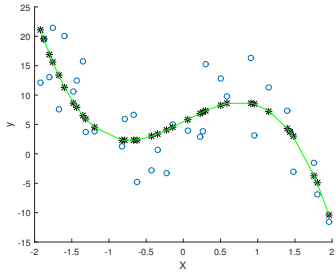
11. 45.28

12.



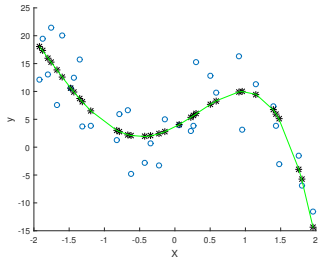
13. 44.93

14.



15. 23.68

16.



17. 20.78

Chapter 6

Overfitting and underfitting: visualisation

6.1 Theoretical Briefing

Same data, different models. It is possible to find several models to fit a given dataset. We saw an example of this in the previous chapter, in which n-th degree polynomial regression was introduced. Each value of n corresponds to a different model, and so it is possible to generate, in principle, infinitely many models to fit the data. Also, in coming chapters we will see other algorithms which can generate even more models. Having such an abundance, how to choose the "best" one?

Evaluation of models. To choose a model, apply performance measures such as mean squared errors, misclassification errors and confusion matrices, and pick out the one with lowest error. Quite straightforward. Nevertheless, this chapter is meant to show you that, most of the time, too much perfection is as bad as too much laxness.

Underfitting. In the first place, let us talk about too much relaxation. We refer here to models that really "take it easy" and give a very lazy prediction about the data, without taking into account particularities and details. The most relaxed model for a classification problem, for example, would be one which predicts the same class for every instance (say, the class "woman" for all of the photos of a given set, regardless of the actual gender of the person depicted.) In Fig 6.1 we show a regression model which predicts the same value of y for every value of x (you will obtain this plot in the Practical.) We say that this kind of models **underfits** the data.

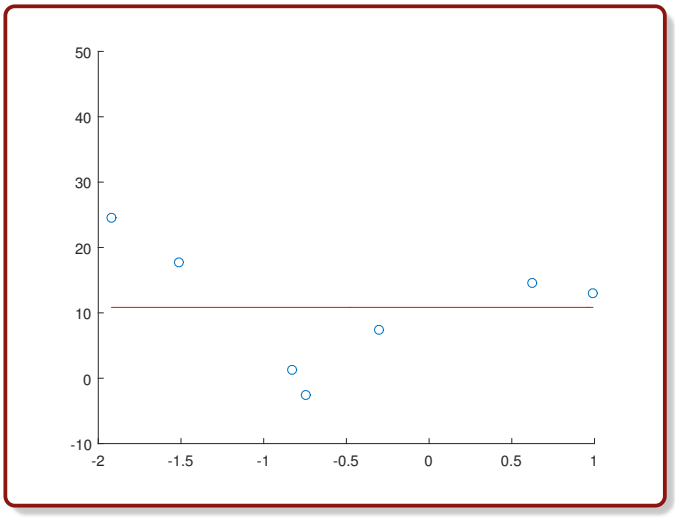


FIGURE 6.1: A model that underfits the data.

Overfitting. You could think, then, that the best model is one which perfectly fits all of the data. In Fig 6.2 we see a linear regression model that attains this (you will obtain this plot in the Practical as well.)

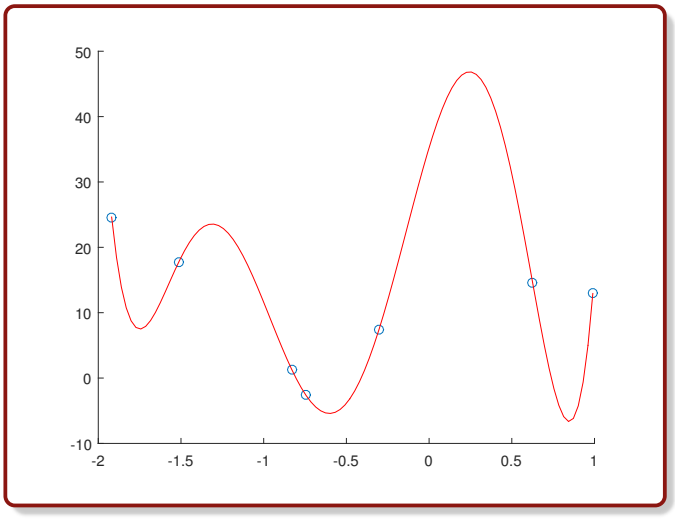


FIGURE 6.2: A model that overfits the data.

Nevertheless, one desired characteristic of a model is its **generalisation capability.** This is, if the model is presented with a new instance, not used pre-

viously in the training phase, it should predict a coherent value of y. Let us show that this is not the case with the model of Figure 6.2. Consider the value of $x = 0.3$. By looking at the plot in Figure 6.2, what is the value of y for this x? Well, it is about 50! Clearly, this is an exaggerated prediction, out of place with respect to the other values. We say that this model does not have a good generalisation capability, and this is because it **overfits** the data.

Dealing with overfitting. In future chapters, we will see how to overcome problems such as overfitting. One possibility is using a validation set to select the best model, a technique that we will use in Chapter 11.

6.2 Exercises

Solve the following exercises. You can use a calculator. Do not use MATLAB here

Let

$$\mathbf{X} = \begin{bmatrix} -2 & -1 & 1 & 2 & 3 & 4 & 5 & 6 & 8 \end{bmatrix}^T$$

and

$$\mathbf{y} = \begin{bmatrix} 1 & 3 & 5 & 4 & 2 & 5 & 7 & 4 & 5 \end{bmatrix}^T$$

be the data for a regression problem.

1. On graph paper, draw a Cartesian plane and plot the data in $\mathbf{X}$ in the horizontal axis and the data in $\mathbf{y}$ in the vertical one.

2. Draw a straight line that fits well to the data (according to your intuition).

3. Draw a curve that overfits the data (again, according to your intuition).

4. In Fig 6.3, we show a plot in MATLAB of the 1^{st}-degree polynomial regression for this data. According to this model, what is the value of $\widehat{y}$ for $x = 7.5$ (approximately)?

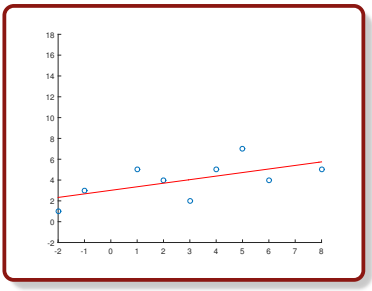


FIGURE 6.3

5. In Fig 6.4, we show a plot in MATLAB of the 8^{th}-degree polynomial regression for this data. According to this model, what is the value of $\widehat{y}$ for $x = 7.5$ (approximately)?

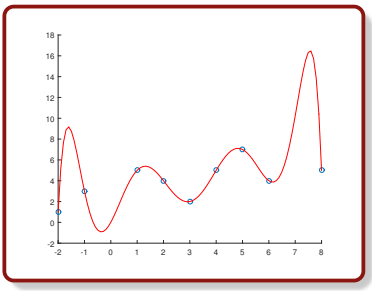


FIGURE 6.4

6.3 Practical in MATLAB

General Objective:

To make the student capable of computing and comparing the solutions given by different models to a given data set.

Specific Objectives:

- The student should be able to compute different solutions for a data set given as example.
- The student should be able to graphically represent the solutions and identify the problems of overfitting and underfitting in these graphical representations.
- The student should be able to estimate the error for each solution and identify the problems of overfitting and underfitting through these performance measures.

Materials:

Computer, MATLAB.

Procedure:

Solve the following exercises in MATLAB

1. Create a script and save it following the format `Pr6_nameLastname.m`. You will write your code for the next numerals in this script.
2. Load into the *Workspace* the matrices contained in the file `matricesPr6.mat`. This file can be found in the folder *Data Sets*.
3. In numeral 2 you loaded a data matrix `X` of one-dimensional vectors and a vector y of the corresponding labels. Use `scatter` to plot the data contained in `X` and `y`. Use the MATLAB command `ylim` to set the limits in the y axis as -10 and 50.
4. By looking at the plot obtained in the previous numeral, make a guess about what the labels (the values of y_i) should be for the "vectors" contained in the matrix `Xnew`. Do not compute anything here, just use your

intuition.

$$\mathbf{X}_{new} = \begin{bmatrix} -1.7500 \\ -1.3000 \\ 0.2500 \\ 0.8500 \end{bmatrix}$$

5. Write a function "`plot_nModel(X,y,n)`", which takes a data matrix `X` of one-dimensional feature vectors, a vector `y` of labels and a parameter `n`; and does not return anything but plots the data in `X` and `y` together with the model corresponding to a n-th degree polynomial regression (see the *Theoretical Briefing* of Chapter 5.)

6. Use your function `plot_nModel(X,y,n)` to obtain plots of the models with:

   a. $n = 0$

   b. $n = 1$

   c. $n = 2$

   d. $n = 3$

   e. $n = 6$

   Set the limits of the y axis as -10 and 50 in all the cases.

7. Write a function "`nModel(X,y,n)`", which takes a data matrix `X` of one-dimensional feature vectors, a vector `y` of labels and a parameter `n`; and returns the weight vector, `w`, corresponding to a n-th degree polynomial regression.

8. Use your function `nModel` to get the vectors `w` and `yhat` for the `Xnew` matrix given in numeral 4, corresponding to the models with:

   a. n = 0

   b. n = 1

   c. n = 2

   d. n = 3

   e. n = 6

   Let us call these vectors `yhat_0`, `yhat_1`, `yhat_2`, `yhat_3` and `yhat_6`, correspondingly. In the *Answers* section, you can see the vectors you should obtain.

9. The author of this book made his own guess about the labels for `Xnew` (see numeral 4). His intuitions are contained in vector $\mathbf{y}_{guess}$.

$$\mathbf{y}_{guess} = \begin{bmatrix} 20 \\ 10 \\ 10 \\ 17 \end{bmatrix}$$

   Compute the mean squared error, with respect to these intuitions, of the predictions contained:

   a. In **yhat_0**
   b. In **yhat_1**
   c. In **yhat_2**
   d. In **yhat_3**
   e. In **yhat_6**

10. Discuss with your classmates about which model was the best and how the phenomena of overfitting and underfitting were unveiled through this practical.

6.4 Answers to selected exercises

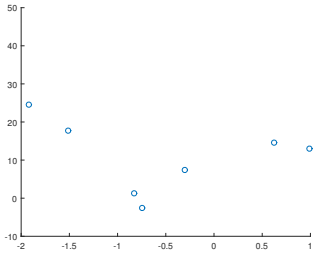
Exercises

4. Approximately, yhat = 5

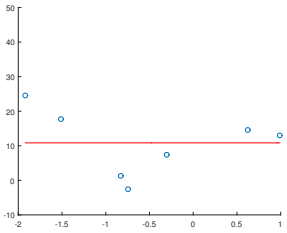
5. Approximately, yhat = 17

Practical

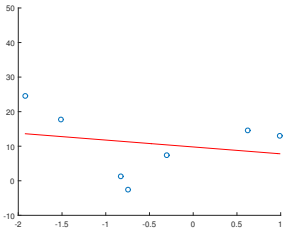
3.



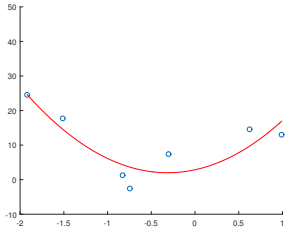
6. (a)



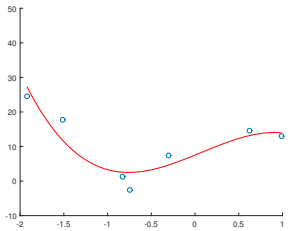
(b)



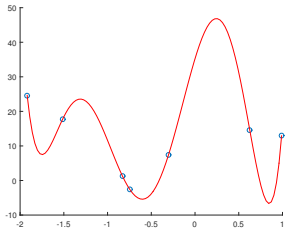
(c)



(d)



(e)



8. (a) `yhat_0` $= \begin{bmatrix} 10.8386 \\ 10.8386 \\ 10.8386 \\ 10.8386 \end{bmatrix}$

(b) `yhat_1` $= \begin{bmatrix} 13.2707 \\ 12.3750 \\ 9.2898 \\ 8.0955 \end{bmatrix}$

(c) `yhat_2` $= \begin{bmatrix} 20.0764 \\ 10.5280 \\ 4.7923 \\ 13.8713 \end{bmatrix}$

(d) `yhat_3` $= \begin{bmatrix} 19.6855 \\ 6.9666 \\ 10.0951 \\ 14.0033 \end{bmatrix}$

(e) `yhat_6` $= \begin{bmatrix} 7.4999 \\ 23.5441 \\ 46.8416 \\ -6.6499 \end{bmatrix}$

9. (a) 30.8249

   (b) 32.6795

   (c) 9.2984

   (d) 4.5723

   (e) 564.0793