

Solution 1 (a)

Step 1

Given that we must predict y and have no knowledge of x , the best predictor for y is the mean of the y values.

Let $y_1 = 1$, $y_2 = 3$, $y_3 = 4$, $y_4 = 6$, $n = 4$.

Calculate mean of the y values.

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i = \frac{y_1 + y_2 + y_3 + y_4}{n} = \frac{1 + 3 + 4 + 6}{4} = 3.5$$

Hence, the best predictor for y is $\bar{y} = 3.5$.

Step 2

Calculate the mean squared error (MSE).

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2 = \frac{(y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 + (y_3 - \bar{y})^2 + (y_4 - \bar{y})^2}{n}$$

$$MSE = \frac{(1 - 3.5)^2 + (3 - 3.5)^2 + (4 - 3.5)^2 + (6 - 3.5)^2}{4} = 3.25$$

Hence, the MSE for the four points is $MSE = 3.25$.

\therefore the best predictor for y is $\bar{y} = 3.5$ with $MSE = 3.25$.

Solution 1 (b)

Step 1

Calculate $y_{prediction}$ using $y = x$ for the following points:

$$(1, 1), (1, 3), (4, 4), (4, 6)$$

$$y_{prediction}(1, 1) = 1$$

$$y_{prediction}(1, 3) = 1$$

$$y_{prediction}(4, 4) = 4$$

$$y_{prediction}(4, 6) = 4$$

Step 2

Calculate the mean squared error (MSE).

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - y_{predicted}(x_i, y_i))^2$$

$$MSE = \frac{(y_1 - y_{predicted}(1, 1))^2 + (y_2 - y_{predicted}(1, 3))^2 + (y_3 - y_{predicted}(4, 4))^2 + (y_4 - y_{predicted}(4, 6))^2}{n}$$

$$MSE = \frac{(1 - 1)^2 + (3 - 1)^2 + (4 - 4)^2 + (6 - 4)^2}{4} = 2$$

\therefore the MSE of the linear function $y = x$ on the points, $(1, 1), (1, 3), (4, 4), (4, 6)$, is 2.

Solution 1 (c)

Step 1

The line that minimizes the MSE is the *line of best fit* is:

$$MSE(a, b) = \frac{1}{n} \sum_{i=1}^n (y_i - (ax_i + b))^2$$

or

$$MSE(a, b) = \frac{(y_1 - (ax_1 + b))^2 + (y_2 - (ax_2 + b))^2 + \dots + (y_n - (ax_n + b))^2}{n}$$

Let $x_1 = 1, y_1 = 1, x_2 = 1, y_2 = 3, x_3 = 4, y_3 = 4, x_4 = 6, y_4 = 6, n = 4$.

Substitute the values into the loss function($MSE(a, b)$):

$$MSE(a, b) = \frac{(y_1 - (ax_1 + b))^2 + (y_2 - (ax_2 + b))^2 + (y_3 - (ax_3 + b))^2 + (y_4 - (ax_4 + b))^2}{4}$$

$$MSE(a, b) = \frac{(1 - (a(1) + b))^2 + (3 - (a(1) + b))^2 + (4 - (a(4) + b))^2 + (6 - (a(4) + b))^2}{4}$$

$$MSE(a, b) = \frac{(1 - (a + b))^2 + (3 - (a + b))^2 + (4 - (4a + b))^2 + (6 - (4a + b))^2}{4}$$

Step 2

The line that minimizes the MSE is the *line of best fit* can be obtained from the following two normal equations:

$$\begin{cases} nb + a \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \\ b \sum_{i=1}^n x_i + a \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i \end{cases}$$

From the above equations, we can compute the slope a and intercept b of the line of best fit using the following equations:

$$a = \frac{n \sum_{i=1}^n x_i y_i - (\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n \sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \quad b = \bar{y} - a\bar{x}$$

Where $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ are the means of x and y respectively.

Substituting the values of x and y into the equations:

$$a = \frac{4((1 \times 1) + (1 \times 3) + (4 \times 4) + (4 \times 6)) - (1 + 1 + 4 + 4)(1 + 3 + 4 + 6)}{4(1^2 + 1^2 + 4^2 + 4^2) - (1 + 1 + 4 + 4)^2}$$

$$a = \frac{4(1 + 3 + 16 + 24) - (10)(14)}{4(1 + 1 + 16 + 16) - (10)^2}$$

$$a = \frac{4(44) - 140}{4(34) - 100}$$

$$a = \frac{176 - 140}{136 - 100}$$

$$a = \frac{36}{36} = 1$$

Substituting the value of a into the equation for b :

$$b = \bar{y} - a\bar{x} = 3.5 - 1(2.5) = 1$$

The line of best fit is $y = x + 1$.

Step 3

Calculate MSE of the line of best fit using equation from **Step 1**:

$$MSE(a, b) = \frac{(1 - (a + b))^2 + (3 - (a + b))^2 + (4 - (4a + b))^2 + (6 - (4a + b))^2}{4}$$

$$MSE(a, b) = \frac{(1 - (1(1) + 1))^2 + (3 - (1(1) + 1))^2 + (4 - (1(4) + 1))^2 + (6 - (1(4) + 1))^2}{4}$$

$$MSE(a, b) = \frac{(1 - (1 + 1))^2 + (3 - (1 + 1))^2 + (4 - (4 + 1))^2 + (6 - (4 + 1))^2}{4}$$

$$MSE(a, b) = \frac{(1 - 2)^2 + (3 - 2)^2 + (4 - 5)^2 + (6 - 5)^2}{4}$$

$$MSE(a, b) = \frac{(1)^2 + (1)^2 + (1)^2 + (1)^2}{4}$$

$$MSE(a, b) = \frac{1 + 1 + 1 + 1}{4} = 1$$

\therefore the line of best fit is $y = x + 1$ with $MSE = 1$.

Solution 2 (a)

Step 1

The loss function is defined as:

$$L(s) = \frac{1}{n} \sum_{i=1}^n (x_i - s)^2$$

Compute the derivative of $L(s)$ with respect to s ($\frac{dL}{ds}$).

$$\frac{dL}{ds} = \frac{1}{n} \sum_{i=1}^n (x_i - s)^2 \frac{d}{ds} = \frac{1}{n} \sum_{i=1}^n (x_i^2 - 2x_i s + s^2) \frac{d}{ds} = -\frac{2}{n} \sum_{i=1}^n (x_i - s)$$

\therefore the derivative of $L(s)$ with respect to s is:

$$\frac{dL}{ds} = -\frac{2}{n} \sum_{i=1}^n (x_i - s)$$

Solution 2 (b)

Step 1

Set the derivative from part (a) to zero to find the value of s :

$$-\frac{2}{n} \sum_{i=1}^n (x_i - s) = 0 \quad (1)$$

$$-\frac{n}{2} \cdot -\frac{2}{n} \sum_{i=1}^n (x_i - s) = 0 \cdot -\frac{n}{2} \quad (2)$$

$$\sum_{i=1}^n x_i - \sum_{i=1}^n s = 0 \quad (3)$$

$$\sum_{i=1}^n x_i - ns = 0 \quad (4)$$

$$\sum_{i=1}^n x_i - ns + ns = 0 + ns \quad (5)$$

$$\sum_{i=1}^n x_i = ns \quad (6)$$

$$\frac{1}{n} \sum_{i=1}^n x_i = s \quad (7)$$

$$\bar{x} = s \quad (8)$$

\therefore the value of s is \bar{x} .

Solution 3

Proof:

We are given a dataset (DS) such that for all data points $(x^{(i)}, y^{(i)})$, where $x^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}$.

If we predict $\hat{y}^{(i)}$ and the true value is $y^{(i)}$, the penalty is the absolute difference:

$$|y^{(i)} - \hat{y}^{(i)}|$$

Let $x^{(i)}$ be the vector of all $x^{(i)} \in DS$ and m be the vector of slope coefficients.

$$x^{(i)} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(n)} \end{bmatrix} \quad m = \begin{bmatrix} m^{(1)} \\ m^{(2)} \\ \vdots \\ m^{(n)} \end{bmatrix}$$

Assume y as a linear function of x , we can express $\hat{y}^{(i)}$ as:

$$\hat{y}^{(i)} = m^\top \cdot x^{(i)} + b$$

where $m \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the parameters of the linear model.

The total penalty on the training set with n points is the sum of the absolute errors over all n data points:

$$\sum_{i=1}^n |y^{(i)} - \hat{y}^{(i)}|$$

Substituting $\hat{y}^{(i)} = m^\top \cdot x^{(i)} + b$ into the equation yields:

$$\sum_{i=1}^n |y^{(i)} - (m^\top \cdot x^{(i)} + b)|$$

\therefore The loss function corresponding to the total penalty on the training set is:

$$L(m, b) = \sum_{i=1}^n |y^{(i)} - (m^\top \cdot x^{(i)} + b)|$$

Solution 4 (a)

Given that x is a picture of an animal and y is the name of the animal. Typically each animal has a unique appearance, then the mapping from x to y is deterministic.

\therefore there is generally little inherent uncertainty in this scenario

Solution 4 (b)

Given that x consists of the dating profiles of two people and y is whether they will be interested in each other. A person's interest in others depends on many observable and unobservable factors.

\therefore there is a significant amount of inherent uncertainty in this scenario.

Solution 4 (c)

Given that x is a speech recording and y is the transcription of the speech into words. Assuming the recording is high quality then the mapping from x to y is deterministic.

\therefore there is generally little inherent uncertainty in this scenario.

Solution 4 (d)

Given that x is the recording of a new song and y is whether it will be a big hit. Whether or not a song is a hit depends on many factors.

\therefore there is a significant amount of inherent uncertainty in this scenario.

Solution 5 (a)

Let, labels $y \in \{-1, 1\}$.

The decision boundary is defined as:

$$\Pr(y = 1 \mid x) = \Pr(y = -1 \mid x)$$

We know that sum of the probabilities over all labels is equal to 1:

$$\Pr(y = 1 \mid x) + \Pr(y = -1 \mid x) = 1$$

From the above two equations, we can see that $\Pr(y = 1 \mid x) = 0.5$ and $\Pr(y = -1 \mid x) = 0.5$.

$\therefore c = 0.5$ is the decision boundary for this classifier.

Solution 5 (b)

The case $\Pr(y = 1 \mid x) = \frac{3}{4}$ represents a region where the model predicts $y = 1$ with higher confidence.

Solution 5 (c)

The case $\Pr(y = 1 \mid x) = \frac{1}{4}$ represents a region where the model predicts $y = -1$ with higher confidence and is symmetrically opposite the decision boundary (a) compared to (b).

Solution 6 (a)

Strategy

The *mystery.dat* dataset contains 101 features. However, only ten of these features are relevant, while the remaining 91 features are simply noise.

∴ I will use Lasso Regression because it forces coefficients to 0 for irrelevant features.

Solution 6 (b)

The ten features identified as relevant are:

[4, 6, 1, 22, 10, 26, 2, 12, 18, 16]

Solution 6 Code

Python Code

```
1  ## import libraries
2  import numpy as np
3  from sklearn.linear_model import LassoCV, Lasso
4
5  ## read mystery.dat
6  data = np.loadtxt('mystery.dat', delimiter=',')
7  x = data[:, :-1]
8  y = data[:, -1]
9
10 ## normalize x based on maximum value in all x data
11 x_norm = x / np.max(np.max(X, axis=0), axis=0)
12
13 ## find best alpha using LassoCV
14 ## test 100 alphas ranging from 1e-4 to 1e4
15 lasso_cv = LassoCV(alphas=np.logspace(-4, 4, 100), cv=5)
16 lasso_cv.fit(x_norm, y)
17 alpha_best = lasso_cv.alpha_
18 print(f"best alpha: {alpha_best}")
19
20 ## fit Lasso model with best alpha
21 lasso = Lasso(alpha=alpha_best)
22 lasso.fit(x_norm, y)
23
24 ## get absolute value of lasso coefficients
25 coefs = np.abs(lasso.coef_)
26
27 ## get indices of the 10 largest coefficients
28 ## note: that the coefficients are sorted in descending order
29 indices = np.argsort(coefs)[::-1][:10]
30
31 ## print the indices (coordinate number) of the 10 largest coefficients
32 print("indices of the 10 largest coefficients:")
33 for i in indices:
34     print(f"coordinate number: {i}, coefficient: {coefs[i]}")
```

Solution 7 (a)

I would choose the 3 features that have the largest |coefficients| for the logistic regression.

∴ I would select top 3 coefficients and the corresponding features: ca (coefficient ≈ 1.78), cp (coefficient ≈ 1.65) and thalach (coefficient ≈ 1.52)

Solution 7 (b)

\therefore the test error for the logistic regression was 0.1845 or 18.45%.

Solution 7 (c)

Comparing error using 5-fold cross-validation, to the models error on the test set was the following:

$$\begin{aligned}\text{log_reg test error} &= 0.1845 \\ \text{mean 5-fold cross-validation error} &= 0.2000 \\ \text{Difference (log_reg - cv)} &= 0.0155\end{aligned}$$

Solution 7 Code

Python Code

```
1  ## import libraries
2  from sklearn.model_selection import train_test_split, cross_val_score
3  from sklearn.linear_model import LogisticRegression
4  from sklearn.metrics import accuracy_score
5  import numpy as np
6  import os
7
8  ## read heart.csv
9  ## note: the first column is the index, so we skip it
10 ## col names are:
11 ## age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,target
12 data = np.loadtxt('heart.csv', delimiter=',', dtype=None, skiprows=1)
13 column_names = os.popen('head -1 heart.csv').read().split(',')
14 x = data[:, :-1]
15 y = data[:, -1]
16
17 ## normalize each x column based on maximum value in column
18 ## note: this will scale each feature [0,1]
19 x_norm = x / np.max(x, axis=0)
20
21 ## split data into training and test sets
22 ## note: train_test_split shuffles the data before splitting
23 ## note: test points = 103 and train points = 200 for test_size=0.339
24 x_train, x_test, y_train, y_test = train_test_split(x_norm, y, test_size=0.339,
25                                                    random_state=33)
26
27 ## fit logistic regression model
28 log_reg = LogisticRegression(max_iter=1000, solver='liblinear')
29 log_reg.fit(x_train, y_train)
30
31 ## predict on test set and calculate accuracy
32 y_pred = log_reg.predict(x_test)
33 test_accuracy = accuracy_score(y_test, y_pred)
34
35 ## get absolute value of logistic regression coefficients
36 coefs = np.abs(log_reg.coef_[0])
37
38 ## get indices of the 3 largest coefficients
39 ## note: that the coefficients are sorted in descending order
40 indices = np.argsort(coefs)[::-1][:3]
41
42 ## top 3 coefficients and features
43 print("top 3 coefficients and features:")
44 for i in indices:
45     print(f"index: {i} coefficient: {coefs[i]}, feature: {column_names[i]}")
46
47 ## log_reg test error
48 test_error = 1 - test_accuracy
49 print("log_reg test error:", test_error)
50
51 ## 5-fold cross-validation error
52 cv_scores = cross_val_score(log_reg, x_train, y_train, cv=5, scoring='accuracy')
53 cv_error = 1 - cv_scores.mean()
54 print("mean 5-fold cross-validation error:", cv_error)
55
56 # compare log_reg test error and 5-fold cross-validation error
57 print(f"Difference (log_reg - cv): {test_error - cv_error:.4f}")
```