Foundations of Machine Learning Homework 1

Laure Dassy

November 7 2021

1 The Learning Problem and Model Selection

1.1 Question 1

answer 5.
The best way to pick a final model to use and estimate its error is to train a new model on the full data set, using the θ you found; use the average CV error as its error estimate. This is because after having used 10-fold cross validation to see how well our model is performing to predict our dataset, we then have to train our model on all the data. Cross validation is used for assessing performance before training our final model. In order to estimate the error of our model, we use the average from the cross validation error.

2 Dimensionality Reduction

2.1 Question 2

a.
Let $\mathbf{M}_{m,n}$ data matrix. The transpose of $\mathbf{M}^T_{n,m}$ is then of dimensions n x m. Then, by matrix multiplication, $\mathbf{M}\mathbf{M}^\top$ has dimensions m x m and $\mathbf{M}^\top\mathbf{M}$ n x n. Thus, the matrices $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ are square.

In addition, we have
$(\mathbf{M}\mathbf{M}^\top)^\top = (\mathbf{M}^\top)^\top\mathbf{M}^\top = \mathbf{M}\mathbf{M}^\top$

and
$(\mathbf{M}^\top\mathbf{M})^\top = \mathbf{M}^\top(\mathbf{M})^\top)^\top = \mathbf{M}^\top\mathbf{M}$

Thus, the matrices $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ are symmetric.

Finally, we know both matrices are composed of real numbers, from which we can infer that the matrices are real.

b.
Let $\lambda \neq 0$ be an eigenvalue of the matrix $\mathbf{M}^\top\mathbf{M}$. This implies that
$\mathbf{M}^\top\mathbf{M}x = \lambda x$ for some $x \neq 0$

$\equiv \mathbf{M}\mathbf{M}^\top(\mathbf{M}x) = \lambda(\mathbf{M}x)$

The equation above allows us to conclude that $\mathbf{M}x$ is an eigenvector of $\mathbf{M}^\top\mathbf{M}$ and its corresponding eigenvalue is λ. We have showed that both matrices share the same non-zero eigenvalues , however their eigenvectors are different (ie. $\mathbf{M}x \neq x$).

When $\lambda = 0$, we will prove that $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ do not share the same zero eigenvalues through an example.

Take a square matrix $\mathbf{A}$

$\mathbf{A} = \begin{bmatrix}1\\0\end{bmatrix}$ We get : $\mathbf{A}\mathbf{A}^\top = \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$ with eigenvalues 1 and 0. And we also have: $\mathbf{A}^\top\mathbf{A} = \begin{bmatrix}1\end{bmatrix}$ with one eigenvalue, 1.

By the use of an example we have proven that the two matrices $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ do not share identical zero eigenvalues nor eigenvectors.

c.
We have $\mathbf{M} = U\Sigma V^\top$ and we know that Σ is a diagonal matrix, so $\Sigma^\top = \Sigma$

We can write that
$\mathbf{M}^\top = ((U\Sigma V)^\top)^\top = V\Sigma U^\top$

We get
$\mathbf{M}^\top\mathbf{M} = U\Sigma V^\top V\Sigma U^\top = V\Sigma^2 V^\top$

Similarly we have $\mathbf{M}^\top\mathbf{M}\mathbf{M}^\top = U\Sigma V^\top V\Sigma U^\top = U\Sigma^2 U^\top$

d. The eigenvalues of $\mathbf{M}^\top\mathbf{M}$ are also the squares of the singular values of M. We see this because we have seen that
$\mathbf{M} = U\Sigma V^\top$

and
$\mathbf{M}^\top \mathbf{M} = V \Sigma^2 V^\top$

3 Model Evaluation

3.1 Question 3

a.
Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$
Algorithm 1 Accuracy = 0.985
Algorithm 2 Accuracy = 0.98

b.
For identifying poisonous mushrooms, algorithm 2 is better. We get a sense of this when comparing the tables: table 1 shows that the algorithm wrongly labeled three poisonous mushrooms as edible vs. algorithm 2 which didn't do the same error. By going further, we conclude that algorithm 2 is best by looking into the sensitivity of the algorithms.
Sensitivity = $\frac{TP}{TP+FN}$
Algorithm 1 Sensitivity = 0.97
Algorithm 2 Sensitivity = 1

Indeed, Algorithm 2 is a more reasonable choice.

4 Regression

4.1 Question 4

The correct answer is the answer 2, increases bias and decreases variance.

If the weights are all larger than 1, then increasing p, λ, the coefficient of penalties increases. This means the the model complexity will be penalized throughout the training phase, forcing the model to become simpler and thus increasing the space for bias to play in. As bias increases, variance decreases, thus answer 2 is correct.

4.2 Question 5

a.
To find the closest form solution of the ridge regression problem we must minimize the given equation by taking the derivative of the arg and setting it equal to 0.

For such, we will first put our cost function in matrix form as below:
$(y - X\theta)^\top (y - X\theta) + \lambda\theta^\top\theta$
$= (y^\top - X\theta)^\top (y - X\theta) + \lambda\theta^\top\theta$
$= (y^\top y - 2\theta^\top X^\top y + \theta^\top X^\top X\theta) + \lambda\theta^\top\theta$

We now find the derivative of the cost function with respect to θ
$\frac{\partial f}{\partial \theta} = 0$
Solving this, we get:
$-2X^T y + 2X^T X\theta + 2\lambda\theta = 0$
With the help of lecture 3 and the Moore-Penrose pseudoinverse generalization we saw in class, we can draw that
$\theta = (X^\top X + \lambda I)^{-1} X^\top y)$

b.
The ridge regression estimator is more robust to overfitting than the least-square's estimator because it introduces a regularization term which penalises parameters that are too large, here θ and shrinks them to 0. Hence, ridge prevents overfit through regularization but also by brining the coefficients of less "useful" variables close to zero where as least-square takes in a large number of independent variables and does not adapt to their "importance".

4.3 Question 6

a.
We know $tanh(\alpha) = 2\sigma(2\alpha) - 1$ and $tanh(\alpha) = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} - e^{-\alpha}}$

We also know that

$\sigma = \frac{1}{1+e^{-x}}$

Substituting, we get

$tanh(\alpha)$

$=2\sigma(2\alpha) - 1$
$=2(\frac{1}{1+e^{-2\alpha}}) - 1$
$=\frac{2}{1+e^{-2\alpha}} - 1$
$=\frac{1-e^{-2\alpha}}{1+e^{-2\alpha}}$
$=\frac{e^{\alpha} - e^{-2\alpha+\alpha}}{e^{\alpha} + e^{-2\alpha+\alpha}}$
$=\frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} - e^{-\alpha}}$

$=tanh(\alpha)$

We showed that the tanh function and the logistic sigmoid function are related.

b.
We will show that a general M-th order polynomial linear combination of logistic functions of the given form are equivalent to a linear combination of tanh functions of the given form noted below:

We have
$y(x, u) = u_0 + \sum_{j=1}^{M} u_j tanh(\frac{x-\mu_j}{2s})$

Substituting again, we get

$y(x, u) = u_0 + \sum_{j=1}^{M} u_j(2s((\frac{x-\mu_j}{s}) - 1)$

$= u_0 + \sum_{j=1}^{M} 2u_j((\sigma(\frac{x-\mu_j}{s})) - \mu_j$

$=u_0 - \sum_{j=1}^{M} u_j + \sum_{j=1}^{M} 2u_j((\sigma(\frac{x-\mu_j}{s}))$

We see in the above equation that when we set $\theta_0 = \sum_{j=1}^{M} u_j and \theta_j = 2u_j$, we get equivalent forms between tanh functions and general M-th order polynomial linear combinations.

4.4 Question 7

We have the following probability distribution: $P_\theta(x) = 2\theta x e^{-\theta x^2}$ where θ is a parameter and x is a positive real number.

To compute the MLE will:

- Compute the log likelihood function
$L(\theta) = LOG(\sum_{i=1}^{n} P_\theta(x_i)) = \sum_{i=1}^{n}(LOG(2) + LOG(\theta) + LOG(x_i - \theta x^2))$

- Find its derivative with respect to θ
$\frac{\partial f}{\partial \theta} = \sum_{i=1}^{n}(\frac{1}{\theta} - x_i^2) = \frac{n}{\theta} - \sum_{i=1}^{n} x_i^2$
- Set the derivative equal to 0 to retrieve the MLE

$\frac{n}{\theta} - \sum_{i=1}^{n} x_i^2 = 0$

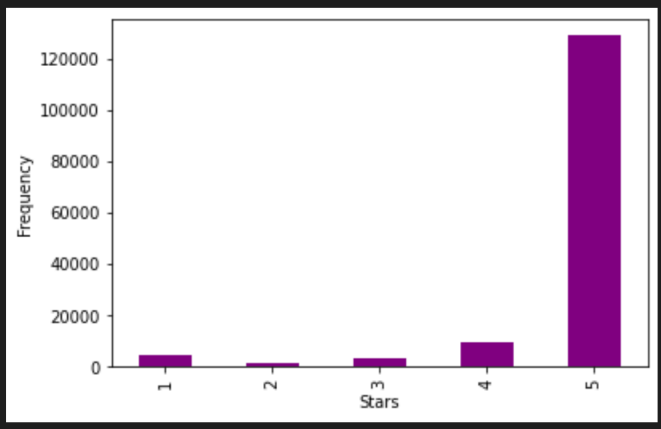
After simplification, we get

$\theta = \frac{n}{\sum_{i=1}^{n} x_i^2}$

4.5 Question 8

a.
The graph below shows the distribution of the ratings within the data set i.e. how frequently levels of ratings are attributed to reviews.

```
<AxesSubplot:xlabel='Stars', ylabel='Frequency'>
```




```
df = pd.read_csv('amazon_reviews_us_Gift_Card_v1_00.tsv', delimiter='\t')
level_ratings = df['star_rating'].value_counts().sort_index()
level_ratings.plot(color=['purple'], kind ='bar', xlabel = 'Stars', ylabel = 'Frequency')
```

✓ 1.9s Python

b.

Find below the code used to train the LR model.

```
def lengthreview(input):
    result =[]
    result = input['review_body']
    result = result.str.split().str.len()
    result = result.fillna(0)
    result= result.to_numpy().astype(float)
    return result

def verifiedpurchase(input):
    return [1 if input['verified_purchase'][i]=='Y' else 0 for i in range (len(input))]
    return result
```

✓ 0.3s Python

```python
# Create a new dataframe with the 2 features and ground truth values
df_lr = pd.DataFrame()
df_lr['star_rating'] = df['star_rating']
df_lr['verified_purchase'] = verifiedpurchase(df)
df_lr['length_review'] = lengthreview(df)


x = df_lr[{'verified_purchase', 'length_review'}]
y = df_lr['star_rating']
model = LinearRegression().fit(x,y)

print('Theta 0 is ' + str(model.intercept_))
print('Theta 1 is  ' + str(model.coef_[1]))
print('Theta 2 is  ' + str(model.coef_[0]))
```

✓ 2.3s Python

```
Theta 0 is 4.854960427901718
Theta 1 is  0.0446965527816378
Theta 2 is  -0.0067940612591762695
```

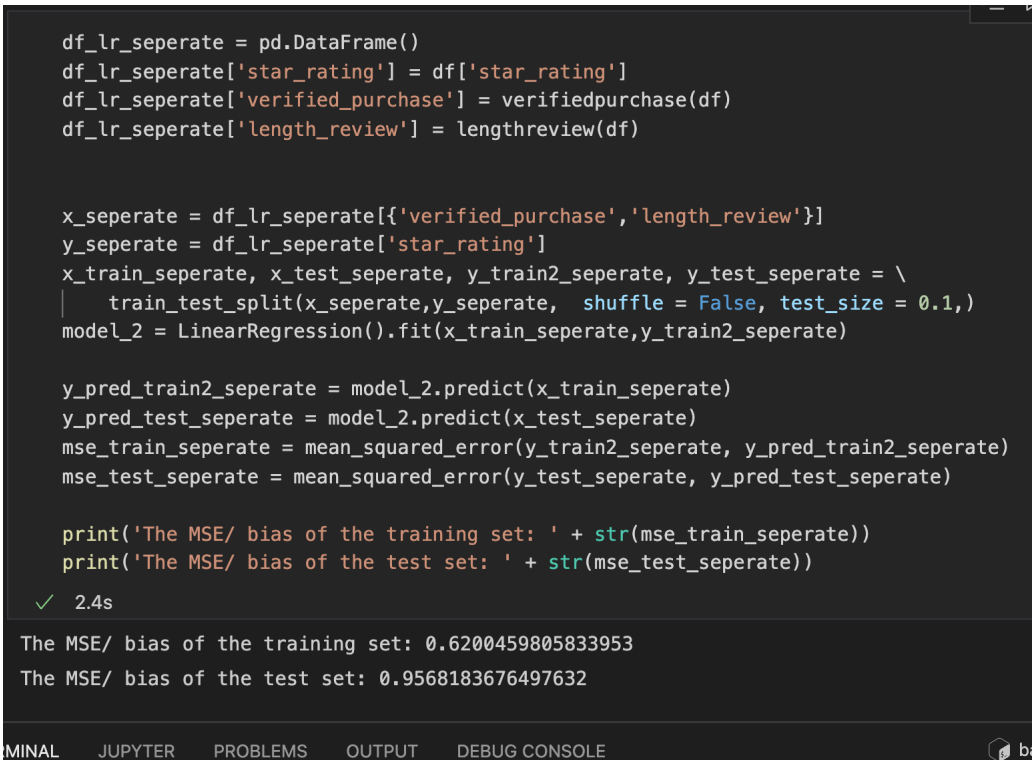
We create two functions to identify the features : "verified purchase" (returns 1 or 0 for Y or no respectively)and "length review" (returns length).

The coefficients we got post training are the following:

$\theta_0 = 4.8549$
$\theta_1 = 0.0447$
$\theta_2 = -0.0068$

These coefficients deduce that our model will use a base average of approximately 4.85 to predict ratings, it will then add or do nothing to that average depending on whether the purchase if verified or not. Finally, if the length review is too long, the model will substract from the average rating a penalty of weight θ_2

c.
At this stage, we take the same data set but we divide into two fractions: the training set with 90 percent of the data and the test set with the remaining 10 percent.
Find below the code:



The MSE of the training set is approximately 0.62 and of the test set is 0.95.

d.

Find below a more accurate model using polynomial features. You will find the training test accuracies below , with the expression of the feature vector we have designed.

```
df_lr['helpful_votes'] = df['helpful_votes']
df_lr['total_votes'] = df['total_votes']
y = df_lr['star_rating']
x = df_lr[['verified_purchase','length_review', \
           'helpful_votes', 'total_votes']]


polynomial_features = PolynomialFeatures(2)
x_polynomial = polynomial_features.fit_transform(x)

x_training, x_test, y_training, y_test = \
    train_test_split(x_polynomial, y, test_size = 0.1, shuffle = False)

model_3 = LinearRegression()
model_3.fit(x_training, y_training)

feature_vec = polynomial_features.get_feature_names()
y_pred_train = model_3.predict(x_training)
y_pred_test = model_3.predict(x_test)
mse_train = mean_squared_error(y_training, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

print('The MSE/ bias of the training set: ' + str(round(mse_train,3)))
print('The MSE/ bias of the test set:: ' + str(round(mse_test,3)))
print('The feature vector is the following: \n' + str(feature_vec))
```

✓ 0.2s

```
The MSE/ bias of the training set: 0.594
The MSE/ bias of the test set:: 1.06
Thee Feature vector:
['1', 'x0', 'x1', 'x2', 'x3', 'x0^2', 'x0 x1', 'x0 x2', 'x0 x3', 'x1^2', 'x1 x2', 'x1 x3', 'x2^2', 'x2 x3', 'x3^2']
```

5 Naive Bayes

5.1 Question 9

Posterior probability of $P(C_1|X_1 = -1, X2 = 1)$

$= \frac{P(X_1=-1|C_1)P(X_2=-1|C_1)P(C_1)}{P(X_1=-1|C_1)P(X_2=-1|C_1)P(C_1)+P(X_1=-1|C_2)P(X_2=1|C_2)P(C_2)}$

$= \frac{0.2*0.5*0.1}{0.2*0.5*0.1+0.6*0.3*0.5}$
$= 0.1$

Posterior probability of $P(C_2|X_1 = -1, X2 = 1)$

$= 1 - P(C_1|X_1 = -1, X2 = 1)$
$=0.9$

5.2 Question 10

a.
This statement is correct because X and Y are conditionally independent given Z.
$\equiv$
$X|Z \cap Y|Z = \emptyset$
b.
This statement is incorrect because the statement
P(X,Y) = P(X)P(Y) is valid only if X and Y are unconditionally independent. Here, X and Y are conditionally independent given Z.