Assignment 1 - Machine Learning

Caio de Próspero Iglesias

December 2020

Contents

I The Learning Problem, Model Selection and Evaluation

I.I Question 1

Answer: 5. Since statistical methods tend to perform worse when trained on fewer observations, we should use the whole data set to train our final model. However, to estimate the error, it is better to use the average CV error, since it does not rely on a specif selection of a training and test set, which could result in misleading conclusions. Therefore, a 10-fold CV tends to estimate well the test error of a model and is commonly used. [1]

I.II Question 2

(a) Bias refers to the error that is introduced by approximating a complex model by a simpler model (underfitting) [1], which is what we have here trying to estimate a quadratic function using a linear model.

(b) First, let's compute the regression coefficient θ. For a single data point x, we will have the pair data-label $(x, 2x^2)$. Since we will estimate it by a linear model with intercept 0 (we only have one point to train here, we cannot determine 2 parameters), we have that $\theta = \frac{\Delta y}{\Delta x} = \frac{2x^2}{x} = 2x$, since $x \neq 0$. Then we have: $h(z) = 2xz$ Moreover, using the definition of bias, we can calculate the bias of $\hat{f}(z) = h(z)$:

$$Bias(\hat{f}(z)) = \mathbb{E}_x[\hat{f}(z) - f(z)] = \mathbb{E}_x[h(z) - g(z)] = \mathbb{E}_x[2xz - 2z^2] = \overset{0}{\cancel{\mathbb{E}_x[x]}}2z - 2z^2 = -2z^2 \tag{1}$$

So, our answer is: $\boxed{Bias(h(z)) = -2z^2}$

(c) By using the definition of the variance, we get:

$$Var(h(z)) = \mathbb{E}_x[(h(z) - \mathbb{E}_x[h(z)])^2] = \mathbb{E}_x[(2xz - \overset{0}{\cancel{\mathbb{E}_x[x]}}2z)^2] = \mathbb{E}_x[4x^2z^2] = 4z^2\,\mathbb{E}_x[x^2] \tag{2}$$

Therefore, we need to calculate $\mathbb{E}_x[x^2]$.

$$\mathbb{E}_x[x^2] = \int_{-1}^{1} \frac{x^2}{2} dx = \frac{x^3}{6}\Big|_{-1}^{1} = \frac{1}{3} \tag{3}$$

Then, we get: $\boxed{Var(h(z)) = \frac{4z^2}{3}}$

(d) The expected test MSE if defined as

$$Er(h, z) = \mathbb{E}_x[(g(z) - h(z))^2] = Var(h(z)) + [Bias(h(z)]^2 + \underbrace{\overset{0}{\cancel{Var(\epsilon)}}}_{noise-free} = \frac{4z^2}{3} + 4z^4 \tag{4}$$

So, for z = 1, we have $Var(h(z)) = \frac{4}{3}$, $Bias(h(z)) = -2$, and then $\boxed{Er(h, z) = \frac{16}{3}}$

II Regression

II.I Question 3

Answer: 2. We are considering a regularized linear regression model of the following form

$$\arg\min_{\theta} \|y - \mathbf{X}\theta\|_2^2 + \lambda \|\theta\|_p^p \tag{5}$$

and we want to study its behavior when p increases ($p \geq 1$). We will consider here that λ is fixed for all p. Therefore, since all the weights (θ_i) are bigger than 1, $\|\theta\|_p^p = \sum_{i=1}^{M} |\theta_i|^p$ will increase as p increases. This means that we will be penalizing more and more the model as p increases. When we penalize the model, we decrease its flexibility, which results in an increase in bias and decrease in variance.

II.II Question 4

(a) We need to minimize the function

$$J(\theta) = \|y - X\theta\|_2^2 + \lambda \|\theta\|_2^2 = (y - X\theta)^T(y - X\theta) + \lambda\theta^T\theta \tag{6}$$

By noticing that $y^T X\theta \in \mathbb{R}$ and, thus, $y^T X\theta = (y^T X\theta)^T = \theta^T X^T y$, we have

$$J(\theta) = \theta^T X^T X\theta - 2y^T X\theta + y^T y + \lambda\theta^T\theta \tag{7}$$

Then, since we want to find the $\hat{\theta}$ that minimizes $J(\theta)$ and since the Ridge Regression problem is strictly convex, we can do

$$\nabla_\theta J(\theta) = 0 \implies \nabla_\theta J(\theta) = 2X^T X\theta - 2X^T y + \lambda 2\theta = 0 \tag{8}$$

And, by developing it further, we have

$$2X^T y = (2X^T X + 2\lambda\mathbb{I})\theta \tag{9}$$

And then, $\boxed{\hat{\theta} = (X^T X + \lambda\mathbb{I})^{-1} X^T y}$

(b) From the last exercise, we have $\nabla_\theta J(\theta) = 2X^T X\theta - 2X^T y + \lambda 2\theta$. By deriving it one more time, we can get the Hessian matrix

$$\boxed{\nabla_\theta^2 J(\theta) = 2X^T X + 2\mathbb{I}\lambda} \tag{10}$$

(c) The ridge regression is an example of a shrinkage method, which aims to shrink the coefficient estimates towards zero. Therefore, as λ increases, the flexibility of the model decreases, which leads to a decrease in variance and an increase in bias. This reduction of the variance and of the flexibility of the model helps it to be more robust to overfitting.

II.III Question 5

(a) We have that

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{11}$$

Therefore, we have

$$2\sigma(2\alpha) - 1 = \frac{2e^{2\alpha}}{1+e^{2\alpha}} - 1 = \frac{2e^{2\alpha} - (1+e^{2\alpha})}{1+e^{2\alpha}} = \frac{e^{2\alpha}-1}{1+e^{2\alpha}} \tag{12}$$

And, then, by multiplying the numerator and denominator by $e^{-\alpha}$, we get

$$2\sigma(2\alpha) - 1 = \frac{e^{\alpha} - e^{-\alpha}}{e^{-\alpha}+e^{\alpha}} = \tanh(\alpha) \tag{13}$$

(b) We know from the first question that $\tanh(\alpha) = 2\sigma(2\alpha) - 1$. So, we also have that $\sigma(2\alpha) = \frac{\tanh(\alpha)+1}{2}$, which we can also write as $\sigma(\beta) = \frac{\tanh(\frac{\beta}{2})+1}{2}$. So, we have

$$y(x,\theta) = \theta_0 + \sum_{j=1}^{M} \theta_j \, \sigma(\frac{x-\mu_j}{s}) = \theta_0 + \sum_{j=1}^{M} \theta_j \, \frac{\tanh(\frac{x-\mu_j}{2s}) + 1}{2} \tag{14}$$

And then, we get

$$y(x,\theta) = \underbrace{\theta_0 + \sum_{j=1}^{M} \frac{\theta_j}{2}}_{u_0} + \sum_{j=1}^{M} \underbrace{\frac{\theta_j}{2}}_{u_j} \tanh(\frac{x-\mu_j}{2s}) \tag{15}$$

And we get, finally, that

$$y(x,u) = u_0 + \sum_{j=1}^{M} u_j \tanh(\frac{x-\mu_j}{2s}) \tag{16}$$

with $\boxed{u_0 = \theta_0 + \sum_{j=1}^{M} \frac{\theta_j}{2}}$ and $\boxed{u_j = \frac{\theta_j}{2}}$

II.IV Question 6

(a) By executing the following code, after importing the important packages, we get the graph representing the distribution, in Figure 1.

```python
file_name = './amazon_reviews_us_Gift_Card_v1_00.tsv'
df = pd.read_csv(file_name, sep='\t')
sns.set_palette("husl")
sns.set_style("whitegrid")
sns.set_context('talk')
plt.title('Distribution Ratings')
plt.ylabel('Frequence')
sns.distplot(df['star_rating'])
```

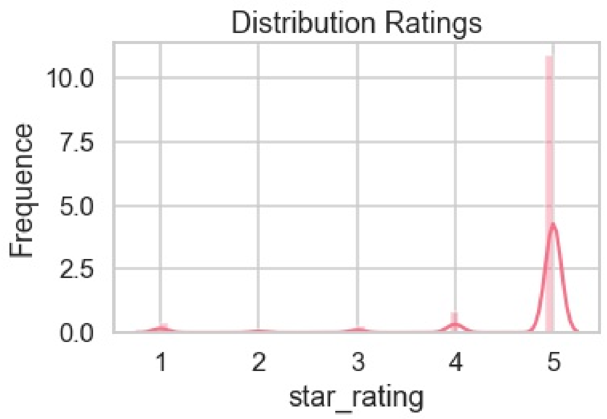

Figure 1: Distribution of Ratings

(b) Here, we use the following code to prepare the data and to train the linear regression model.

```python
df = df.dropna() #drop nan values
df['length_review'] = df['review_body'].apply(
    lambda x: len([i for i in re.split('\.|\s|,',x) if i != '']))
df['verified_purchase'] = df['verified_purchase'].astype('category')
X = df[['verified_purchase', 'length_review']]
X['verified_purchase'] = X['verified_purchase'].apply(lambda x:
                                                       1 if x == 'Y'
                                                       else 0)
y = df['star_rating']
model = LinearRegression()
model.fit(X,y)
results = model.intercept_, model.coef_
```

We then get $\boxed{\theta_0 \approx 4.85}$, $\boxed{\theta_1 \approx 0.04}$ and $\boxed{\theta_2 \approx -0.0067}$

We have then that the verified purchase is usually associated with a better review and that the bigger the length of the review, the worse the rating. However, the parameters found are quite small and we should thus perform a statistical test to verify its real significance.

(c) We will now use some functions from sklearn to calculate the Mean Squared Errors on the training and test sets.

```
X_train, X_test, y_train, y_test = train_test_split(
                                        X,y, test_size = 0.1,
                                        shuffle = False)
model_separate = LinearRegression()
model_separate.fit(X_train, y_train)
y_pred_train = model.predict(X_train)
y_pred_test = model_separate.predict(X_test)
mse_train = MSE(y_pred_train, y_train)
mse_test = MSE(y_pred_test, y_test)
```

We then get $\boxed{MSE_{test} \approx 0.95}$ and $\boxed{MSE_{train} \approx 0.62}$, which makes sense, since $MSE_{train} < MSE_{test}$

(d) After preparing the data, we can execute the following code to analyse the MSE for the test and train datasets.

```
list_errors_train = []
list_errors_test = []
for i in range (1,4):
    print('Calculating polynomial with degree = {}'.format(i))
    poly = PolynomialFeatures(i)
    model = LinearRegression()
    X_train_poly = poly.fit_transform(X_train)
    X_test_poly = poly.fit_transform(X_test)
    model.fit(X_train_poly, y_train)
    y_pred_train = model.predict(X_train_poly)
    y_pred_test = model.predict(X_test_poly)
    test_error = MSE(y_pred_test,y_test)
    train_error = MSE(y_pred_train, y_train)
    print('Test error: {:.2}'.format(test_error))
    print('Train error: {:.2}'.format(train_error))
    list_errors_train.append(train_error)
    list_errors_test.append(test_error)

plt.plot(range(1,4), list_errors_train, label = 'train')
plt.plot(range(1,4), list_errors_test, label = 'test')
plt.legend()
```

Which gives the following output:

```
Calculating polynomial with degree = 1
Test error: 0.95
Train error: 0.61
Calculating polynomial with degree = 2
Test error: 1.1
Train error: 0.59
Calculating polynomial with degree = 3
```

```
Test error: 1.2e+01
Train error: 0.58
```

which we can represent graphically by

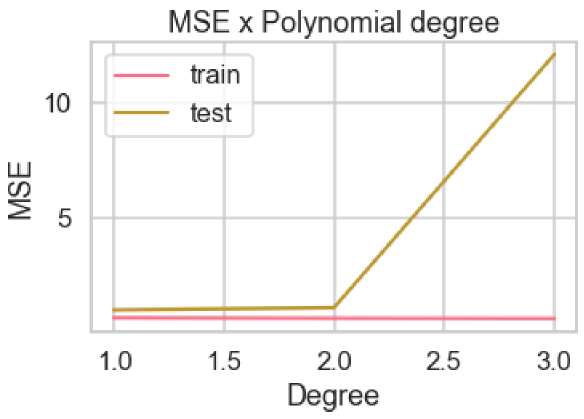

Figure 2: Mean squared error X Degree

The expression for the feature vector, for degree = 2, for example, is

```
['1','x0','x1','x2','x3','x0^2','x0x1,'x0 x2','x0 x3','x1^2',
'x1x2','x1x3','x2^2','x2 x3','x3^2']
```

We can thus conclude that the best option here is to use a linear model, since increasing the complexity of the model increases the MSE_{test} (overfitting). Moreover, adding the other features do not really improve the model's performance, which indicate they do not influence the rating.

III Generative Models

III.I Question 7

We have the following probability distribution function, and we draw m $i.i.d$ samples from it.

$$\mathbb{P}_\theta(x) = 2\theta e^{-\theta x^2} \tag{17}$$

And then the likelihood function is $\mathcal{L}(\theta, \mathbf{x}) = \prod_{i=1}^m 2\theta x_i e^{-\theta x_i^2}$. So, the maximum likelihood estimator is

$$\hat{\theta} = \arg\max_\theta \prod_{i=1}^m 2\theta x_i e^{-\theta x_i^2} \tag{18}$$

Since the function $x \mapsto \ln(x)$ is increasing, we can maximize $\ln(\mathcal{L})$ instead, because it has some nice properties. We then have

$$ln(\mathcal{L}) = \sum_{i=1}^m (\ln(2\theta) + \ln(x_i) - \theta x_i^2) \tag{19}$$

Then, we can find $\hat{\theta}$ by doing

$$\partial_\theta \ln(\mathcal{L}) = 0 \implies \sum_{i=1}^m (\frac{1}{2\theta}.2 - x_i^2) = \frac{m}{\theta} - \sum_{i=1}^m x_i^2 = 0 \tag{20}$$

So, we get $\boxed{\hat{\theta} = \frac{m}{\sum_{i=1}^m x_i^2} = \frac{1}{\frac{1}{m}\sum_{i=1}^m x_i^2}}$. One might also verify that $\partial_\theta^2 \ln(\mathcal{L})) < 0$ and that it is indeed a maximization problem.

IV Nearest Neighbors

IV.I Question 8

(a) Answer: 1 and 2. On the KNN, the smaller the k, the more flexible the model, which results in high variance and low bias. The variance comes from the fact that we consider a small number of points to do our estimation and thus the estimation will depend highly on single data points. However, when k increases, we have a smoother and less variable fit, since we consider the results from a bigger number of points. Nevertheless, this may add some bias, since the average of several points might mask some properties from the function $f(X)$ we are trying to estimate. That is why it is important to have a good balance between bias and variance, in order to get the smallest MSE_{test} as possible. [1]

(b) Answer: 1. (the error is 0). The training error for a KNN with $k = 1$ will always be 0, since we will take as a prediction the nearest neighbor which will be the point itself , because, by the definition of a distance, $d(x, x) = 0$, which is the smallest distance possible ($\forall x, y \; d(x, y) \geq 0$). Then, we will always use as a prediction the real value and this will result in a training error of 0. However, this result is obviously misleading, since it is a clear example of overfitting and would most probably perform badly in the test data set.

References

[1] T. Hastie G. James, D. Witten and R. Tibshirani. *An Introduction to Statistical Learning with applications in R.* Springer, 2017.