IDS 702: Module 1.7

Mean squared error and cross validation

Dr. Olanrewaju Michael Akande

Mean squared error

- One particularly useful metric for measuring model fit (especially when the goal is prediction) is the mean squared error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

- This value will be small when our predictions $\hat{y}_i$ are close to the true y_i's. Some analysts and data scientists will often report the root mean squared error (RMSE) instead, which is simply the square root of MSE.

- While it may be useful to calculate within-sample MSE using the same dataset that was used to fit the model (usually referred to as training data), it is often more useful to calculate out-of-sample MSE using a different dataset (usually referred to as test data).

- In other words, while it may be great to know that our model fits the data used in fitting it well, it would be even better to see that our model also fits new or future data well.

- This is essentially asking the question: what does our model tell us about what might happen in the future?

Mean squared error

- If we have a large amount of data, we can split our sample into training and test datasets.
- The test dataset should contain new observations $(y_{1i}, \boldsymbol{x}_{1i})$ that are not represented in the training dataset $(y_{0i}, \boldsymbol{x}_{0i})$.
- Then the test MSE or out-of-sample MSE is given by

$$\text{MSE}_{\text{test}} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_{1i} - \hat{y}_{1i})^2.$$

  where $\hat{y}_{1i}$ is the predicted response for a new observation in the test dataset **using the model fitted using the training dataset**, and n_{test} is the number of new observations in the test dataset.

- The smaller the MSE (whether in-sample or out-of-sample), the better.
- However, because "small" can be relative depending on the scale of y, we often use MSEs when comparing different models (again, particularly when the goal is prediction). We will see this later.

Training and test data

- Using test data is often important because of the problem of overfitting.
- Overfitting arises when the model is working too hard to find the perfect predictions in the training data and is not broadly generalizable because it ends up picking up patterns that are just reflecting random error.
- We generally expect the test MSE to be somewhat larger than the training MSE because our model has been developed to minimize the training MSE.
- Overfitting refers to a situation in which a different model (generally a simpler one) fit to the training data would result in a smaller test MSE (indicating better out-of-sample prediction).
- We may be able to identify this problem when comparing the out-of-sample MSEs of different models (including the parsimonious models).
- Note that in small datasets, the random split of the data can have considerable impact on the results; out-of-sample MSEs can differ greatly depending on which random sample we take.

Training and test data

Let's explore this concept using the last fitted regression. We will use three different random splits. For the first split, we have

```
#set the seed to ensure we can replicate the same result
set.seed(123)
train_index <- sample(nrow(wages),round(0.7*nrow(wages)),replace=F)
train <- wages[train_index,]
test <- wages[-train_index,]
regwagecsquares_train <- lm(bsal~sex+seniorc+agec+agec2+educc+experc+experc2,data=train)
y_test_pred <- predict(regwagecsquares_train,test)
temp <- cbind(test$bsal,y_test_pred);
colnames(temp) <- c("Truth","Predicted"); temp[1:5,]
```

```
##    Truth Predicted
## 1   5040  5705.432
## 2   6300  6254.975
## 3   6000  6376.807
## 10  6900  6548.043
## 11  6900  6103.975
```

```
testMSE <- mean((test$bsal - y_test_pred)^2); testMSE
```

```
## [1] 273782.4
```

```
sqrt(testMSE)
```

```
## [1] 523.2422
```

Training and test data

For the second split, we have

```
#now change the seed
set.seed(1234)
train_index <- sample(nrow(wages),round(0.7*nrow(wages)),replace=F)
train <- wages[train_index,]
test <- wages[-train_index,]
regwagecsquares_train <- lm(bsal~sex+seniorc+agec+agec2+educc+experc+experc2,data=train)
y_test_pred <- predict(regwagecsquares_train,test)
temp <- cbind(test$bsal,y_test_pred);
colnames(temp) <- c("Truth","Predicted"); temp[1:5,]
```

```
##    Truth Predicted
## 1   5040  5705.884
## 7   8100  6390.656
## 11  6900  6136.419
## 12  5400  5795.275
## 13  6000  6385.950
```

```
testMSE <- mean((test$bsal - y_test_pred)^2); testMSE
```

```
## [1] 328104.3
```

```
sqrt(testMSE)
```

```
## [1] 572.8039
```

Training and test data

For the final split, we have

```
#change the seed one more time
set.seed(12345)
train_index <- sample(nrow(wages),round(0.7*nrow(wages)),replace=F)
train <- wages[train_index,]
test <- wages[-train_index,]
regwagecsquares_train <- lm(bsal~sex+seniorc+agec+agec2+educc+experc+experc2,data=train)
y_test_pred <- predict(regwagecsquares_train,test)
temp <- cbind(test$bsal,y_test_pred);
colnames(temp) <- c("Truth","Predicted"); temp[1:5,]
```

```
##    Truth Predicted
## 4   6000  5354.919
## 6   6840  5754.380
## 8   6000  5672.324
## 18  5280  4682.765
## 21  5400  5008.557
```

```
testMSE <- mean((test$bsal - y_test_pred)^2); testMSE
```

```
## [1] 199045.4
```

```
sqrt(testMSE)
```

```
## [1] 446.145
```

K-FOLD CROSS-VALIDATION

- This train/test method of model validation is often referred to as cross-validation. In general, one can use other metrics instead of just the MSE.
- K-fold cross-validation is a type of cross-validation that aims to address the issue of sensitivity of results to particular random data splits.
- Specifically, under K-fold cross-validation, split the data into K mutually-exclusive groups, called folds.
- For the k^{th} fold, with $k = 1, \ldots, K$, fit the model on all the remaining data excluding that k^{th} fold (that is, all the other folds combined) and use the k^{th} fold as the test or validation set.
- Repeat this k times, so that each fold has a turn as the validation set. Obtain the $\text{MSE}_{\text{test}}^{(k)}$ for each k, and summarize the error using

$$\text{Avg.MSE} = \frac{1}{K} \sum_{k=1}^{K} \text{MSE}_{\text{test}}^{(k)}.$$

LEAVE-ONE-OUT CROSS-VALIDATION

- A special case of K-fold cross-validation is the Leave-one-out cross-validation, in which $K = n$ (very computationally intensive except in special cases).
- Test error estimates using $k = 5$ or $k = 10$ have been shown to have good statistical properties, motivating these common choices.
- In the case of least squares, we can get an estimate of the average MSE from leave-one-out cross-validation using a simple formula (sadly, this does not hold in most models) based on the fit of only one model!
- The estimate is

$$\text{Avg.MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2.$$

where h_{ii} is the leverage score of observation i.

How would high leverage points affect Avg.MSE in this case?

Final notes

- Again, after fitting your model, model assessment and validation is A MUST!
- In this class and outside of it, you should always assess and validate your models!
- You will write your own code for doing k-fold cross validation in class.
- We will look at other metrics for validating models later in the class when we get to other models.
- For example, the MSE may not be the best metric to look at when dealing with binary outcomes. Or can it still be useful? We will see!
- Over the next few modules, we will explore methods for model selection and including interaction effects in MLRs.

What's next?

Move on to the readings for the next module!