기상청

[STEP5 분석모형 검증]

모형 검증
(K-fold Cross-validation)

기상청

기상기후
빅데이터 분석 플랫폼

[분석교육] 교차 검증(K-fold Cross-validation)

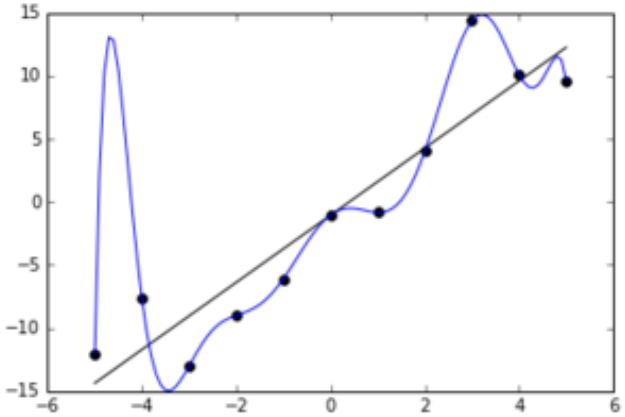
1. 반복문 활용 함수
2. 예측 모형 검증 함수
3. 분류 모형 검증 함수
4. 모형 검증 관련 함수

[분석 교육] 교차 검증(K-fold Cross-validation) (1/2)

전체 데이터를 사용하여 모형을 구축할 경우, 학습한 데이터에는 잘 적합하지만 새로운 데이터가 주어졌을 때, 성능이 좋지 않을 가능성이 있다. 이러한 현상을 과적합이라고 하는데, 과적합을 방지하기 위해 데이터를 학습 데이터(Training Data)와 테스트 데이터(Test Data)로 나누어 교차 검증을 수행한다. 교차 검증(K-fold Cross-validation)은 데이터를 훈련 데이터와 검증 데이터로 나누어 모델링 및 평가하는 작업을 K회 반복함으로써 과적합을 방지하는 검증 기법이다.

● 모형의 과적합(Overfitting)

- 과적합(Overfitting)이란 모형이 학습데이터에 과도하게 적합하는 현상을 의미한다.



〈그림1〉 과적합 예시

- 예를 들어, 〈그림 1〉의 경우, 주어진 데이터는 검은색 직선의 선형 관계로 적합할 수 있다.
- 하지만 파란색 곡선으로 표현된 모형은 주어진 데이터의 노이즈까지 모두 모형에 반영하며 적합하였다.
- 파란색 모형이 주어진 데이터에 더 정확하게 적합 했지만, 일반적인 데이터의 분포에 맞게 적합한 것은 검은색 선형 모형이다.
- 이 때, 파란색 모형의 경우를 과적합이 발생했다고 한다.

- 과적합이 발생하면 현재의 데이터에 대해서는 잘 설명할 수 있을지 모르나, 새로운 데이터가 들어 올 경우 모형의 예측 성능이 떨어질 수 있다.

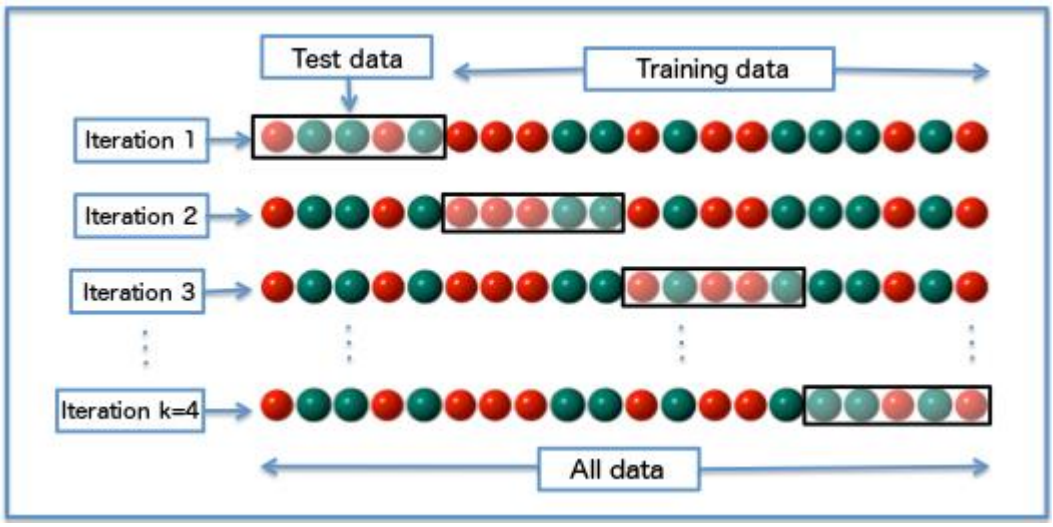
● 교차 검증

- 과적합을 방지하기 위해 훈련 데이터와 테스트 데이터를 분리하여 모형을 구축하는 방법 중 가장 자주 사용하는 기법으로, 데이터를 다수의 조각으로 나누어 훈련과 테스트를 반복하는 기법이다.

[분석 교육] 교차 검증(K-fold Cross-validation) (2/2)

K-fold Cross-validation

- 교차 검증 기법 중, 가장 흔하게 사용하는 기법으로, 훈련 데이터와 검증 데이터로 나누어 모형을 구축하고 평가하는 과정을 K번 반복하는 기법이다.



〈그림2〉 K-fold Cross-validation

- 예를 들어, k =4 로 설정한 K-fold Cross-validation 수행 방법은 다음과 같다.
  - 데이터를 4등분하여 D1, D2, D3, D4로 분할
  - D1을 검증 데이터, 그 외 데이터(D2, D3, D4)를 훈련 데이터로 하여 모형을 구축
  - 검증 데이터 D1을 사용하여 모형의 성능을 평가
  - D2를 검증 데이터, 그 외 데이터(D1, D3, D4)를 훈련 데이터로 하여 모형을 구축
  - 검증 데이터 D2를 사용하여 모형의 성능을 평가
  - 위와 같은 방법으로 D3, D4 데이터까지 반복 수행
- R에서 교차 검증을 제공하는 함수는 cvTools::cvFolds이며, 데이터를 랜덤으로 분할하여 교차 검증을 수행한다.
- 만약 데이터의 속성을 고려하여 데이터를 분할하고자 할 때는 데이터 속성을 기준으로 데이터를 분할 한 뒤, 분할한 길이만큼 반복문으로 모형 구축과 평가를 반복 수행함으로써, 교차 검증을 수행할 수 있다. (ex. 연도별, 관측지점별)

참고 자료 : 위키백과, R을 이용한 데이터 처리&분석 실무, 서민구, 2014

1. 반복문 활용 함수(1/3)

- **For 반복문**

  - 변수 i에 값을 할당하면서 i의 길이만큼 블록 안의 문장을 반복 수행

▪ **Usage**

```
for(i in data){
   i를 포함한 반복 수행 할 문장
 }
```

▪ **Examples**

```
STNList <- unique(DATA$STN_ID) # STN별 리스트 생성
perform <- list(); varimp <- list(); mape <- list(); # 결과리스트 초기화

for(i in 1: length(STNList)){
 stn <- STNList[i]
 print(paste0("STN_ID : ", stn, "...computing"))
 train <- data.hex[data.hex$STN_ID != stn,]
 valid <- data.hex[data.hex$STN_ID == stn,]

 m <- h2o.randomForest(y = y,
                       x = varList,
                       training_frame = train,
                       ntrees = 512,
                       max_depth = 20,
                       sample_rate = 0.3,
                       mtries = -1,
                       seed=1)
 #result
 perform[[i]] <- data.frame(mse = h2o.mse(m) ,r2 =h2o.r2(m))
 varimp[[i]] <- h2o.varimp(m)

 print(paste0("STN_ID : ",stn, "...predicting"))
 #predict
 pred<- h2o.predict(m, newdata = valid)
 Yhat <- as.data.frame(pred$predict)
 Y <- as.data.frame(valid[,y])
 tmp <- cbind(Y,Yhat)
 names(tmp) <- c("SUM_SML_EV","predict")
 mape[[i]]      <-      data.frame(STN_ID      =      stn,      smape_100      =
 cal_smape_100(tmp$SUM_SML_EV,tmp$predict))
}
```

1. 반복문 활용 함수(2/3)

```
> # STN별 리스트 생성
> STNList <- unique(DATA$STN_ID)
> # 결과리스트 초기화
> perform <- list(); varimp <- list(); mape <- list();
> #--------MODEL RUNNING
> for(i in 1: length(STNList)){
+   stn <- STNList[i]
+   print(paste0("STN_ID : ", stn, "...computing"))
+   train <- data.hex[data.hex$STN_ID != stn,]
+   valid <- data.hex[data.hex$STN_ID == stn,]
+
+   m <- h2o.randomForest(y = y,
+                         x = varList,
+                         training_frame = train,
+                         ntrees = 512,
+                         max_depth = 20,
+                         sample_rate = 0.3,
+                         mtries = -1,
+                         seed=1)
+   #result
+   perform[[i]] <- data.frame(mse = h2o.mse(m) ,r2 =h2o.r2(m))
+   varimp[[i]] <- h2o.varimp(m)
+   print(paste0("STN_ID : ",stn, "...predicting"))
+   #predict
+   pred<- h2o.predict(m, newdata = valid)
+   Yhat <- as.data.frame(pred$predict)
+   Y <- as.data.frame(valid[,y])
+   tmp <- cbind(Y,Yhat)
+   names(tmp) <- c("SUM_SML_EV","predict")
+   mape[[i]] <- data.frame(STN_ID = stn, smape_100 = cal_smape_100(tmp$SUM_SML_EV,tmp$predict))
+ }

[1] "STN_ID : 95...computing"
  |======================================================================| 100%
[1] "STN_ID : 95...predicting"
  |======================================================================| 100%
[1] "STN_ID : 101...computing"
  |======================================================================| 100%
[1] "STN_ID : 101...predicting"
  |======================================================================| 100%
[1] "STN_ID : 104...computing"
  |======================================================================| 100%
[1] "STN_ID : 104...predicting"
  |======================================================================| 100%
[1] "STN_ID : 105...computing"
  |======================================================================| 100%
[1] "STN_ID : 105...predicting"
  |======================================================================| 100%
```

〈반복 수행 중…〉

1. 반복문 활용 함수(3/3)

- **cbind**

  - 행열 형태의 벡터, 매트릭스, 데이터 프레임을 열로 합침

  ▪ **Usage**

  cbind(x, y)

  - x,y : 합치고자 하는 행열 형태의 벡터, 매트릭스, 데이터 프레임

  ▪ **Examples**

```
str(Prediction); str(Test_Y); #cbind 할 데이터프레임
Prediction <- cbind(Prediction, Test_Y)
str(Prediction)
```

```
>   str(Prediction);    str(Test_Y); #cbind 할 데이터프레임
Classes 'data.table' and 'data.frame':  22 obs. of  2 variables:
 $ predict: num  6483 6513 6753 6481 6773 ...
 $ y      : num  6581 6581 6581 6463 6463 ...
 - attr(*, ".internal.selfref")=<externalptr>
Classes 'data.table' and 'data.frame':  22 obs. of  1 variable:
 $ y: num  6581 6581 6581 6463 6463 ...
 - attr(*, ".internal.selfref")=<externalptr>
>   Prediction <- cbind(Prediction, Test_Y)
>   str(Prediction)
Classes 'data.table' and 'data.frame':  22 obs. of  3 variables:
 $ predict: num  6483 6513 6753 6481 6773 ...
 $ y      : num  6581 6581 6581 6463 6463 ...
 $ y      : num  6581 6581 6581 6463 6463 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

- **data.table::rbindlist**

  - 리스트를 데이터 테이블로 변환

  ▪ **Usage**

  rbindlist(l, fill = FALSE, …)

  - l : 데이터 테이블이나 데이터 프레임 형태를 포함한 리스트
  - fill : 컬럼이 없을 경우 NA 값으로 처리

  ▪ **Examples**

```
head(GLM_Validation_Result,2)
GLM_Validation_Result_list <- rbindlist(GLM_Validation_Result)
head(GLM_Validation_Result_list,2)
```

```
> head(GLM_Validation_Result,2)
[[1]]
  Year      R2  MAPE
1 2001 0.30792 27.72

[[2]]
  Year      R2  MAPE
1 2002 0.29653 5.765

> GLM_Validation_Result_list <- rbindlist(GLM_Validation_Result)
> head(GLM_Validation_Result_list,2)
   Year      R2   MAPE
1: 2001 0.30792 27.720
2: 2002 0.29653  5.765
```

2. 예측 모형 검증 함수

❖ 예측 모형 평가 지표

- **Mean Squared Error(MSE)**
  - 오차의 제곱에 평균을 취한 것, MSE가 작을수록 추정값의 정확성이 높음

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

- **R^2(R-square)**
  - 모형의 결정계수로 모형이 자료에 적합한 정도를 재는 척도
  - 데이터의 총 변동(SST)중 회귀선으로 설명되는 부분(SSR)이 차지하는 비율을 구함

$$R^2 \equiv 1 - \frac{SS_{\mathrm{res}}}{SS_{\mathrm{tot}}}.$$

h2o::h2o.mse

- 예측 모형의 평균 제곱 오차(Mean Squared Error:MSE)를 산출

- **Usage**
  - object : H2O 모델 객체
- **Examples**

h2o.mse(fit)

```
> h2o.mse(fit)
[1] 2.055386
```

h2o::h2o.r2

- 예측 모형의 R^2 (R-square)를 산출

- **Usage**
  - object : H2O 모델 객체
- **Examples**

h2o.r2(fit)

```
> h2o.r2(fit)
[1] 0.4762723
```

3. 분류 모형 검증 함수

❖ 분류 모형 평가 지표

- **AUC(Area Under the Curve)**
  - ROC curve(1-Specificity X Sensitivity) 아래의 면적
  - 분류 모형의 성능 평가 지표로 0.5부터 1까지 값을 가지며, 1에 가까울 수록 발생/미발생 건에 대한 모형의 예측 성능이 좋음을 의미
- **분류 결과표(Confusion Matrix)**
  - 실제 결과(Y)를 모형이 예측한 결과(Y hat, $\hat{Y}$)와 비교한 표

Actual \ Predict	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

● h2o::h2o.auc

- 분류 모형의 AUC(Area Under the Curve)를 산출

▪ **Usage**
- object : H2O 모델 객체

▪ **Examples**
h2o.auc(fit)

```
> h2o.auc(Fit)
[1] 0.9751626
```

● h2o::h2o.confusionMatrix

- 분류 모형의 분류 결과표(Confusion Matrix)를 산출

▪ **Usage**
- object : H2O 모델 객체
- newdata : 실제 결괏값이 들어있는 H2OFrame 구조의 데이터

▪ **Examples**
h2o.confusionMatrix(fit, newdata = test)

```
> h2o.confusionMatrix(Fit, newdata = test)
Confusion Matrix (vertical: actual; across: predicted)  for max f1 @ threshold = 0.355359203169576:
           0    1    Error          Rate
0      26316  531 0.019779   =531/26847
1        515 1183 0.303298    =515/1698
Totals 26831 1714 0.036644  =1046/28545
```

4. 모형 검증 관련 함수

- **h2o::h2o.varimp**

  - 모형의 변수 중요도를 산출

  ▪ **Usage**
  - object : H2O 모델 객체

  ▪ **Examples**
  h2o.varimp(fit)

```
> h2o.varimp(fit)
Variable Importances:
    variable relative_importance scaled_importance percentage
1     TS_AVG     18395944.000000          1.000000   0.470819
2 CA_TOT_AVG      6520207.000000          0.354437   0.166876
3      WS_2m      5535983.500000          0.300935   0.141686
4         HT      3874688.000000          0.210627   0.099167
5     SUM_RN      2541166.250000          0.138137   0.065038
6 SD_TOT_MAX      2204213.250000          0.119821   0.056414
```

- **h2o::h2o.predict**

  - H2O모형으로부터 예측값을 산출함

  ▪ **Usage**
  - object : H2O 모델 객체
  - newdata : 예측을 위한 변수를 포함한 H2O 데이터

  ▪ **Examples**
  pred<- h2o.predict(fit, data.hex)
  Yhat <- as.data.frame(pred$predict)
  Y <- as.data.frame(data.hex[,y])
  head(cbind(Y,Yhat))

```
> pred<- h2o.predict(fit, data.hex)
 |=============================================================| 100%
> Yhat <- as.data.frame(pred$predict)
> Y <- as.data.frame(data.hex[,y])
> head(cbind(Y,Yhat))
  SUM_SML_EV   predict
1        1.1 1.2315366
2        1.8 1.4720945
3        0.6 1.1986245
4        0.3 0.7349381
5        0.7 1.3843723
6        2.6 2.2545784
```

본 문서의 내용은 기상청의 날씨마루(http://big.kma.go.kr) 내
R 프로그래밍 교육 자료입니다.