XGBoost模型Boosting過程剖析說明

指導教授：中山財管所 王昭文 教授
簡報製作：中山財管所 蘇彥庭 研究助理
2019/09/23

摘要

- 預測數值-比較XGBoost直接預測與疊代預測結果
- 預測分類-比較XGBoost直接預測與疊代預測結果

預測數值

比較XGBoost直接預測與疊代預測結果

預測數值-比較XGBoost直接預測與疊代預測結果

- 整理數據資料集，此資料集由datarium套件提供

```
######################################## CASE1. 預測數值 ########################################
rm(list = ls()); gc()
# 讀取數據(此為數值資料)
data("marketing", package = "datarium")
trainData <- marketing[c(1:100), !(colnames(marketing) %in% c("sales"))]
trainLabel <- marketing[c(1:100), c("sales")]
valdData <- marketing[c(101:150), !(colnames(marketing) %in% c("sales"))]
valdLabel <- marketing[c(101:150), c("sales")]
testData <- marketing[c(151:200), !(colnames(marketing) %in% c("sales"))]
testLabel <- marketing[c(151:200), c("sales")]
dtrain <- xgb.DMatrix(as.matrix(trainData), label = trainLabel)
dvald <- xgb.DMatrix(as.matrix(valdData), label = valdLabel)
dtest <- xgb.DMatrix(as.matrix(testData), label = testLabel)
```

預測數值-比較XGBoost直接預測與疊代預測結果

- 建立watchlist及固定參數組合

```
# 建立watchlist
watchlist <- list(eval = dvald, train = dtrain)

# 建立參數
params <- list(max_depth = 2, eta = 0.1, objective = "reg:squarederror")
```

目標函數為預測值

- 直接跑XGBoost模型10次疊代

```
# 測試方法: XGBoost模型直接訓練10次疊代
bst <- xgb.train(params = params, data = dtrain, nrounds = 10, base_score = 0.3, watchlist = watchlist)

# 預測結果
directPredcitResult <- predict(bst, dtest)
```

默認值為0.5，此處為測試參數功用故設為0.3

儲存直接跑XGBoost模型的預測結果，
待會與疊代XGBoost模型進行比較

預測數值-比較XGBoost直接預測與疊代預測結果

- base_score參數說明(XGBoost官網)
- https://xgboost.readthedocs.io/en/latest/parameter.html

base_score [default=0.5]

- The initial prediction score of all instances, global bias
- For sufficient number of iterations, changing this value will not have too much effect.

- base_score為預測Y的起始值設定

預測數值-比較XGBoost直接預測與疊代預測結果

- 接下來跑疊代模型
- 疊代模型參考自XGBoost套件在Github上的R程式範例寫法
- https://github.com/dmlc/xgboost/blob/master/R-package/R/xgb.Booster.R

預測數值-比較XGBoost直接預測與疊代預測結果

- 疊代模型訓練寫法

```
# 測試方法: 分開訓練10次
saveModelList <- list()
iter <- 1
for(iter in c(1:10)){

  bst <- xgb.train(params = params,
                   data = dtrain,
                   nrounds = 1,
                   base_score = ifelse(iter == 1, 0.3, 0),
                   watchlist = watchlist)

  # 儲存模型
  saveModelList[[iter]] <- bst

  # Note: we need the margin value instead of transformed prediction in set_base_margin
  # do predict with output_margin=TRUE, will always give you margin values before logistic transformation
  ptrain <- predict(bst, dtrain, outputmargin = TRUE)
  pvald  <- predict(bst, dvald, outputmargin = TRUE)

  # set the base_margin property of dtrain and dvald
  # base_margin is the base prediction we will boost from
  setinfo(dtrain, "base_margin", ptrain)
  setinfo(dvald, "base_margin", pvald)
}
```

每次只跑一個疊代

第一次疊代設定和直接跑XGBoost模型的base_score相同，但第二次開始即要設為0，否則預測結果會不相同

```
# 每次迴圈只跑一次疊代
# 第一次疊代設置和原XGB模型相同 第二次以後設置固定為0
```

儲存每次疊代學到的模型

outputmargin在預測"值"時不論設T或F皆不會有影響

重新設定訓練集及驗證集資料的Y，此處為dtrain及dvald為物件導向變數，所以修改後即會自動更新

預測數值-比較XGBoost直接預測與疊代預測結果

- 疊代模型預測寫法

```
# 預測結果
predictResult <- sapply(saveModelList, predict, dtest)
loopPredcitResult <- apply(predictResult, 1, "sum")
```

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$
$$\dots$$
$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

- predictResult為測試集樣本在每個疊代模型的$f_t(x_i)$

每個疊代模型

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	1.852174	1.4037051	1.1787677	1.6981392	1.016154	1.0676212	0.8494730	1.1839185	0.7299657	0.6323897
2	1.611353	1.1353444	1.1787677	1.0627800	1.016154	0.8366575	0.7747688	0.7186059	0.7299657	0.2728137
3	1.852174	1.4037051	1.8577474	1.6981392	1.016154	1.0676212	0.8494730	1.1839185	0.7299657	0.6323897
4	2.658177	2.1292946	1.8577474	1.6981392	1.569384	1.6406832	1.3570374	0.7186059	1.1880263	0.5602651
5	1.852174	1.4037051	1.8577474	1.6981392	1.016154	1.0676212	0.8494730	1.1839185	0.7299657	0.6323897

樣本

列加總即為最終樣本預測結果

預測數值-比較XGBoost直接預測與疊代預測結果

- 比較直接跑XGBoost模型與疊代XGBoost模型預測結果

```
> # 比較XGBoost模型 直接訓練10次疊代 與 分開訓練10次 結果
> head(directPredcitResult, 20)
 [1] 11.612309  9.337211 12.291289 15.377360 12.291289  5.139397  8.910927  9.999363  5.139397 10.133844 11.074872  8.910927 11.612309
[14] 15.377360  9.624662 10.187180  7.017957 10.187180 12.291289 10.187180
> head(loopPredcitResult, 20)
 [1] 11.612308  9.337211 12.291288 15.377360 12.291288  5.139397  8.910926  9.999362  5.139397 10.133844 11.074871  8.910926 11.612308
[14] 15.377360  9.624662 10.187179  7.017957 10.187179 12.291288 10.187179
> # 小結: 直接跑10次與分開訓練10次之預測結果相同
```

- 由上可看出兩模型預測數值皆相同

預測分類

比較XGBoost直接預測與疊代預測結果

預測分類-比較XGBoost直接預測與疊代預測結果

- 整理數據資料集，此資料集由xgboost套件提供

```
######################################## CASE2. 預測類別 ########################################
rm(list = ls()); gc()
# 讀取數據(此為預測類別資料)
data(agaricus.train, package = "xgboost")
data(agaricus.test, package = "xgboost")
dtrain <- xgb.DMatrix(agaricus.train$data[c(1:3000), ], label = agaricus.train$label[c(1:3000)])
dvald <- xgb.DMatrix(agaricus.train$data[c(3001:nrow(agaricus.train$data)), ],
                     label = agaricus.train$label[c(3001:nrow(agaricus.train$data))])
dtest <- xgb.DMatrix(agaricus.test$data, label = agaricus.test$label)

# 建立watchlist
watchlist <- list(eval = dvald, train = dtrain)

# 建立參數
params <- list(max_depth = 2, eta = 0.1, objective = "binary:logistic")
```

目標函數為預測分類

預測分類-比較XGBoost直接預測與疊代預測結果

- 直接跑XGBoost模型10次疊代

```
# 測試方法: XGBoost模型直接訓練10次疊代
bst <- xgb.train(params = params, data = dtrain, nrounds = 10, base_score = 0.9, watchlist = watchlist)

# 預測結果
directPredcitResult <- predict(bst, dtest)
```

默認值為0.5，此處為測試參數功用故設為0.9
base_score在分類模型時只能設定在0到1之間

預測分類-比較XGBoost直接預測與疊代預測結果

- 疊代模型訓練寫法

```
# 測試方法: 分開訓練10次
saveModelList <- list()
iter <- 1
for(iter in c(1:10)){

  bst <- xgb.train(params = params,
                   data = dtrain,
                   nrounds = 1,                                  # 每次迴圈只跑一次疊代
                   base_score = ifelse(iter == 1, 0.9, 0.5),     # 第一次疊代設置和原XGB模型相同 第二次以後設置固定為0.5
                   watchlist = watchlist)

  # 儲存模型
  saveModelList[[iter]] <- bst

  # Note: we need the margin value instead of transformed prediction in set_base_margin
  # do predict with output_margin=TRUE, will always give you margin values before logistic transformation
  ptrain <- predict(bst, dtrain, outputmargin = TRUE)
  pvald  <- predict(bst, dvald, outputmargin = TRUE)

  # set the base_margin property of dtrain and dvald
  # base margin is the base prediction we will boost from
  setinfo(dtrain, "base_margin", ptrain)
  setinfo(dvald, "base_margin", pvald)
}
```

第一次疊代設定和直接跑XGBoost模型的base_score相同，但第二次開始即要設為0.5，否則預測結果會不相同

outputmargin在預測分類時一定要設為T

預測分類-比較XGBoost直接預測與疊代預測結果

- 疊代模型預測寫法

```
# 預測結果
predictResult <- sapply(saveModelList, predict, dtest, outputmargin = TRUE)
loopPredcitResult <- apply(predictResult, 1, "sum")
LogisticTransformFunction <- function(x) 1/(1+exp(-x))
loopPredcitResult <- LogisticTransformFunction(loopPredcitResult)
```

logistic轉換函數

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$
$$\cdots$$
$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

每個疊代模型

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
1	1.241400	-0.44498536	-0.3153623	-0.2547780	-0.2193866	-0.1950608	-0.1770722	-0.1638919	-0.1534734	-0.1446817
2	2.302459	-0.03480467	0.1041977	0.1027183	-0.2193866	0.1043791	0.1028897	-0.1638919	0.1037525	0.1022966
3	1.241400	-0.44498536	-0.3153623	-0.2547780	-0.2193866	-0.1950608	-0.1770722	-0.1638919	-0.1534734	-0.1446817
4	1.241400	-0.44498536	-0.3153623	-0.2547780	-0.2193866	-0.1950608	-0.1770722	-0.1638919	-0.1534734	-0.1446817
5	1.241400	-0.03480467	-0.3153623	-0.2547780	-0.2193866	-0.1950608	-0.1770722	-0.1638919	-0.1534734	-0.1446817

樣本

列加總需要做logistic轉換才會是最終樣本預測結果

$$P(t) = \frac{1}{1 + e^{-t}}$$

加總後代入

預測分類-比較XGBoost直接預測與疊代預測結果

- 比較直接跑XGBoost模型與疊代XGBoost模型預測結果

```
> # 比較XGBoost模型 直接訓練10次疊代 與 分開訓練10次 結果
> head(directPredcitResult, 20)
 [1] 0.3042180 0.9244643 0.3042180 0.3042180 0.3972082 0.3108401 0.9244643 0.3042180 0.9244643 0.3042180 0.9244643 0.3042180 0.3042180
[14] 0.3042180 0.3042180 0.3042180 0.3042180 0.9244643 0.3972082 0.3042180
> head(loopPredcitResult, 20)
 [1] 0.3042180 0.9244644 0.3042180 0.3042180 0.3972082 0.3108400 0.9244644 0.3042180 0.9244644 0.3042180 0.9244644 0.3042180 0.3042180
[14] 0.3042180 0.3042180 0.3042180 0.3042180 0.9244644 0.3972082 0.3042180
> # 小結: 直接跑10次與分開訓練10次之預測結果相同
```

- 由上可看出兩模型預測數值皆相同