Training options

Jeffrey Leek

May 18, 2016

SPAM Example

```
library(caret); library(kernlab); data(spam)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)
training <- spam[inTrain,]
testing <- spam[-inTrain,]
modelFit <- train(type ~.,data=training, method="glm")
```

Train options

```
args(train.default)
```

```
## function (x, y, method = "rf", preProcess = NULL, ..., w
##     metric = ifelse(is.factor(y), "Accuracy", "RMSE"), m
##         c("RMSE", "logLoss"), FALSE, TRUE), trControl =
##     tuneGrid = NULL, tuneLength = 3)
## NULL
```

Metric options

Continous outcomes: * $RMSE$ = Root mean squared error * $RSquared$ = R^2 from regression models

Categorical outcomes: * $Accuracy$ = Fraction correct * $Kappa$ = A measure of concordance

trainControl

```
args(trainControl)
```

```
## function (method = "boot", number = ifelse(grepl("cv", m
##     10, 25), repeats = ifelse(grepl("cv", method), 1, nu
##     p = 0.75, search = "grid", initialWindow = NULL, hor
##     fixedWindow = TRUE, verboseIter = FALSE, returnData
##     returnResamp = "final", savePredictions = FALSE, cla
##     summaryFunction = defaultSummary, selectionFunction
##     preProcOptions = list(thresh = 0.95, ICAcomp = 3, k
##     sampling = NULL, index = NULL, indexOut = NULL, inde
##     timingSamps = 0, predictionBounds = rep(FALSE, 2), s
##     adaptive = list(min = 5, alpha = 0.05, method = "gls
##     trim = FALSE, allowParallel = TRUE)
## NULL
```

trainControl resampling

- *method*
- *boot* = bootstrapping
- *boot632* = bootstrapping with adjustment
- *cv* = cross validation
- *repeatedcv* = repeated cross validation
- *LOOCV* = leave one out cross validation
- *number*
- For boot/cross validation
- Number of subsamples to take
- *repeats*
- Number of times to repeate subsampling
- If big this can *slow things down*

Setting the seed

- It is often useful to set an overall seed
- You can also set a seed for each resample
- Seeding each resample is useful for parallel fits

seed example

```
set.seed(1235)
modelFit2 <- train(type ~.,data=training, method="glm")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or
```

seed example

```
set.seed(1235)
modelFit3 <- train(type ~.,data=training, method="glm")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or

## Warning: glm.fit: fitted probabilities numerically 0 or
```

Further resources

- Caret tutorial
- Model training and tuning