Practical Machine Learning Week 2 : The caret package lecture 2.6 - Covariates Creation

Jeff Leek, notes by Bruno Fischer Colonimos

02 juin 2017

Contents

1 Lecture 2.5 : Basic Preprocessing

Covariates = predictors = features

1.1 Two levels of covariate creation

Covariates are sometimes called predictors and sometimes called features. They are the variables that you will actually include in your model that you're going to be using to combine them to predict whatever outcome that you care about.

- Level 1: From raw data to covariate : image, text.... ==> variables that describe the raw data
- Level 2: Transforming tidy covariates

```
library(kernlab);data(spam)
spam$capitalAveSq <- spam$capitalAve^2
```

1.2 Level 1, Raw data -> covariates

- Depends heavily on application
- The balancing act is summarization vs. information loss
- Examples:
  - Text files: frequency of words, frequency of phrases (Google ngrams), frequency of capital letters.
  - Images: Edges, corners, blobs, ridges (computer vision feature detection)
  - Webpages: Number and type of images, position of elements, colors, videos (A/B Testing)
  - People: Height, weight, hair color, sex, country of origin.
- The more knowledge of the system you have the better the job you will do.

- When in doubt, err on the side of more features
- Can be automated, but use caution!

1.3 Level 2, Tidy covariates -> new covariates

- More necessary for some methods (regression, svms) than for others (classification trees).
- Should be done only on the training set
- The best approach is through exploratory analysis (plotting/tables)
- New covariates should be added to data frames

1.4 Load example data

```r
library(ISLR); library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

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

```r
data(Wage)
inTrain <- createDataPartition(y=Wage$wage,
                               p=0.7, list=FALSE)
training <- Wage[inTrain,]
testing <- Wage[-inTrain,]
```

1.5 Common covariates to add, *dummy variables* : function *dummyVars()*

Basic idea - convert factor variables to indicator variables : function *dummyVars()* of the caret package

```r
table(training$jobclass)
```

```
##
##  1. Industrial 2. Information
##           1078           1024
```

```r
dummies <- dummyVars(wage ~ jobclass,data=training) # function dummyVars() ==> returns a model object
head(predict(dummies,newdata=training)) # use the object with predict
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 231655                      1                       0
## 161300                      1                       0
## 155159                      0                       1
## 450601                      1                       0
## 228963                      0                       1
## 81404                       0                       1
```

1.6 Removing zero covariates (no variability at all): function nearZeroVar()

```
nsv <- nearZeroVar(training,saveMetrics=TRUE) # function nearZeroVar() of the caret package
nsv # dataframe:
```

```
##            freqRatio percentUnique zeroVar   nzv
## year        1.090062    0.33301618   FALSE FALSE
## age         1.246377    2.85442436   FALSE FALSE
## sex         0.000000    0.04757374    TRUE  TRUE
## maritl      3.147505    0.23786870   FALSE FALSE
## race        8.753769    0.19029496   FALSE FALSE
## education   1.336049    0.23786870   FALSE FALSE
## region      0.000000    0.04757374    TRUE  TRUE
## jobclass    1.052734    0.09514748   FALSE FALSE
## health      2.538721    0.09514748   FALSE FALSE
## health_ins  2.299843    0.09514748   FALSE FALSE
## logwage     1.036585   19.41008563   FALSE FALSE
## wage        1.036585   19.41008563   FALSE FALSE
```

==> eliminate Sex and Region from the features

1.7 Spline basis

Sometimes, you want to be able to fit curvy lines, and one way to do that is with a basis functions, and so you can find those, for example, in the splines package, and so one thing that you can do is create this, the bs function will create a polynomial variable. So in this case, we pass at a single variable, in this case, the training set, we take the age variable, and we say we want a third degree polynomial for this variable. So when you do that, you essentially get, you'll get a three-column matrix out. So this is now three new variables. variable 1 corresponds to age, the actual age values scaled for computational purposes (?? exact meaning ???) variable 2 corresponds to age^2 variable 3 corresponds to age^3

```
library(splines)
bsBasis <- bs(training$age,df=3)
head(bsBasis, 10)
```

```
##                1         2          3
##  [1,] 0.0000000 0.0000000 0.00000000
##  [2,] 0.4163380 0.3211750 0.08258786
##  [3,] 0.4308138 0.2910904 0.06556091
##  [4,] 0.4241549 0.3063341 0.07374710
##  [5,] 0.4403553 0.2596967 0.05105149
##  [6,] 0.3355376 0.4074385 0.16491558
##  [7,] 0.4163380 0.3211750 0.08258786
##  [8,] 0.4261690 0.1482327 0.01718640
##  [9,] 0.4333314 0.1637030 0.02061445
## [10,] 0.4443582 0.2275981 0.03885821
```

See also: ns(),poly()

1.8 Fitting curves with splines

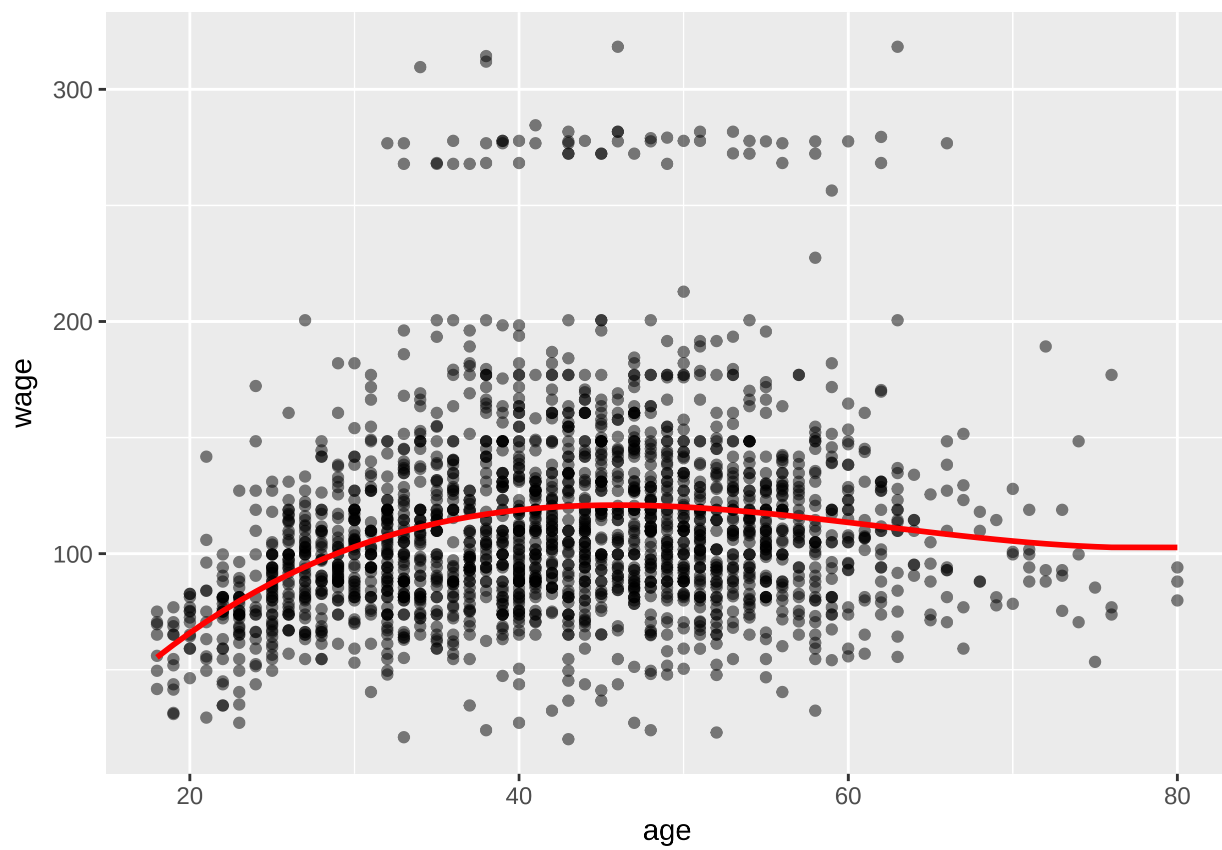
```
# linear model using the splines predictors
lm1 <- lm(wage ~ bsBasis, data=training)

# plot(training$age,training$wage,pch=19,cex=0.5)
# points(training$age, predict(lm1,newdata=training), type = "p", col="red",pch=19,cex=0.5) #????

library(ggplot2)

smoothdf <- data.frame(age = training$age, wage = predict(lm1,newdata=training))

ggplot(training, aes(age, wage)) +
        geom_point( alpha = 0.5) +
        geom_line(data = smoothdf, aes(age, wage), size = 1, color = "red" )
```



1.9 Splines on the test set

```
# regenerate features for the test set, based on those of the training set.
bstest <- predict(bsBasis,age=testing$age)
head(bstest)
```

```
##                1         2          3
## [1,] 0.0000000 0.0000000 0.00000000
## [2,] 0.4163380 0.3211750 0.08258786
```

```
## [3,] 0.4308138 0.2910904 0.06556091
## [4,] 0.4241549 0.3063341 0.07374710
## [5,] 0.4403553 0.2596967 0.05105149
## [6,] 0.3355376 0.4074385 0.16491558
```

1.10 Notes and further reading

1.10.1 Level 1 feature creation (raw data to covariates)

- Science is key. Google "feature extraction for [data type]"
- Err on overcreation of features
- In some applications (images, voices) automated feature creation is possible/necessary
- http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf

1.10.2 Level 2 feature creation (covariates to new covariates)

- The function preProcess in caret will handle some preprocessing.
- Create new covariates if you think they will improve fit
- Use exploratory analysis on the training set for creating them
- Be careful about overfitting!
- preprocessing with caret
- If you want to fit spline models, use the gam method in the caret package which allows smoothing of multiple variables. More on feature creation/data tidying in the Obtaining Data course from the Data Science course track.