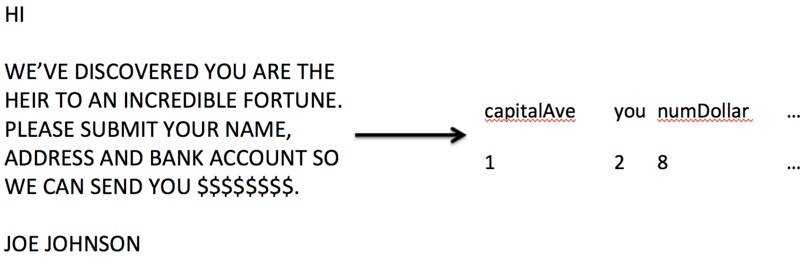
Covariate creation

Jeffrey Leek

May 18, 2016

Two levels of covariate creation

Level 1: From raw data to covariate



Level 2: Transforming tidy covariates

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

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

Load example data

```
library(ISLR); library(caret); data(Wage);
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

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

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

Common covariates to add, dummy variables

Basic idea - convert factor variables to indicator variables

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

```
##
##  1. Industrial 2. Information
##           1083           1019
```

```
dummies <- dummyVars(wage ~ jobclass,data=training)
head(predict(dummies,newdata=training))
```

```
##        jobclass.1. Industrial jobclass.2. Information
## 86582                       0                       1
## 11443                       0                       1
## 450601                      1                       0
## 377954                      0                       1
## 302778                      0                       1
## 305706                      1                       0
```

Removing zero covariates

```
nsv <- nearZeroVar(training,saveMetrics=TRUE)
nsv
```

```
##              freqRatio percentUnique zeroVar   nzv
## year          1.063636    0.33301618   FALSE FALSE
## age           1.109589    2.90199810   FALSE FALSE
## sex           0.000000    0.04757374    TRUE  TRUE
## maritl        3.161572    0.23786870   FALSE FALSE
## race          8.352657    0.19029496   FALSE FALSE
## education     1.443497    0.23786870   FALSE FALSE
## region        0.000000    0.04757374    TRUE  TRUE
## jobclass      1.062807    0.09514748   FALSE FALSE
## health        2.520938    0.09514748   FALSE FALSE
## health_ins    2.199391    0.09514748   FALSE FALSE
## logwage       1.062500   19.45765937   FALSE FALSE
## wage          1.062500   19.45765937   FALSE FALSE
```

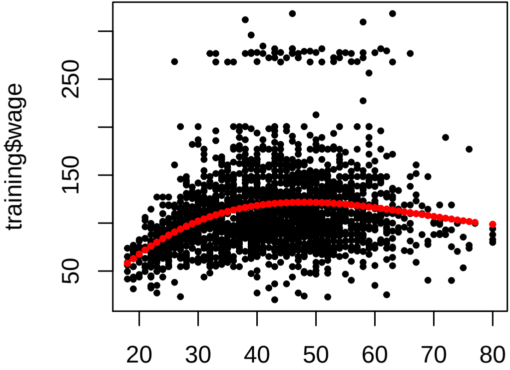
Spline basis

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

```
##                 1            2            3
##     [1,] 0.236850055 0.0253767916 9.063140e-04
##     [2,] 0.362525595 0.3866939680 1.374912e-01
##     [3,] 0.424154946 0.3063341278 7.374710e-02
##     [4,] 0.377630828 0.0906313987 7.250512e-03
##     [5,] 0.416337988 0.3211750193 8.258786e-02
##     [6,] 0.426168977 0.1482326877 1.718640e-02
##     [7,] 0.444358195 0.2275981001 3.885821e-02
##     [8,] 0.306334128 0.4241549461 1.957638e-01
##     [9,] 0.349346279 0.3975319727 1.507880e-01
##    [10,] 0.442218287 0.1953987782 2.877966e-02
##    [11,] 0.362525595 0.3866939680 1.374912e-01
##    [12,] 0.275519452 0.4362391326 2.302373e-01
##    [13,] 0.442218287 0.1953987782 2.877966e-02
```

Fitting curves with splines

```
lm1 <- lm(wage ~ bsBasis,data=training)
plot(training$age,training$wage,pch=19,cex=0.5)
points(training$age,predict(lm1,newdata=training),col="red'
```



Splines on the test set

```r
predict(bsBasis,age=testing$age)
```

```
##                1            2            3
##     [1,] 0.236850055 0.0253767916 9.063140e-04
##     [2,] 0.362525595 0.3866939680 1.374912e-01
##     [3,] 0.424154946 0.3063341278 7.374710e-02
##     [4,] 0.377630828 0.0906313987 7.250512e-03
##     [5,] 0.416337988 0.3211750193 8.258786e-02
##     [6,] 0.426168977 0.1482326877 1.718640e-02
##     [7,] 0.444358195 0.2275981001 3.885821e-02
##     [8,] 0.306334128 0.4241549461 1.957638e-01
##     [9,] 0.349346279 0.3975319727 1.507880e-01
##    [10,] 0.442218287 0.1953987782 2.877966e-02
##    [11,] 0.362525595 0.3866939680 1.374912e-01
##    [12,] 0.275519452 0.4362391326 2.302373e-01
##    [13,] 0.442218287 0.1953987782 2.877966e-02
##    [14,] 0.444093854 0.2114732637 3.356718e-02
##    [15,] 0.443086838 0.2436977611 4.467792e-02
```

Notes and further reading

- 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`
- 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