Lecture 7: Data Preprocessing and Feature Engineering

Alan Lee

Department of Statistics
STATS 784 Lecture 7

August 14, 2017

Outline

Today's agenda

In this lecture we discuss the importance of selecting, transforming and imputing the features used in prediction. We will cover

- Scaling
- Transforming
- Data cleaning and imputation
- Feature selection
- Feature engineering

We will use the Boston housing data as a running example. Note: References to APM refer to *Applied Predictive Modeling*: Ch 3 of APM covers the material of this lecture.

Scaling

Some of the prediction methods we have discussed work better when the features have similar magnitudes. For example

- In Neural networks, the starting values are chosen randomly on an interval around zero. This works better if the features have similar magnitudes. Also, the regularization penalty will apply equally to all variables if they have similar magnitudes.
- In ridge regression and the lasso the same argument applies (these will be discussed in Lecture 10.)
- In unsupervised learning, there are techniques that benefit from having features with similar magnitudes (e.g. principal components.)

It is usual to scale the continuous variables by subtracting off means and dividing by standard deviations, so that the scaled data has mean zero and variance one. Note this only applies to continuous variables! If the data set contains a mixture of continuous variables and factors we need to split it into two parts. The R function `scale` can be used to scale the continuous variables:

Scaling: R code

```
library(MASS)
data(Boston)

# make variable chas into a factor
Boston$chas = factor(Boston$chas)
numericCols = unlist(lapply(Boston, is.numeric))
BostonCont = Boston[, numericCols]
BostonFactor =Boston[, !numericCols]
names(BostonFactor) = names(Boston)[!numericCols]
BostonScaled = data.frame(as.data.frame(scale(BostonCont)),
        BostonFactor)
```

Symmetrizing

It is also advantageous to transform the variables so that they have approximately symmetric distributions. The easiest way to do this is using Box-Cox transformations: we transform a variable x to $x^{(\lambda)}$ using the equation

$$x^{(\lambda)} = \frac{x^\lambda - 1}{\lambda}.$$

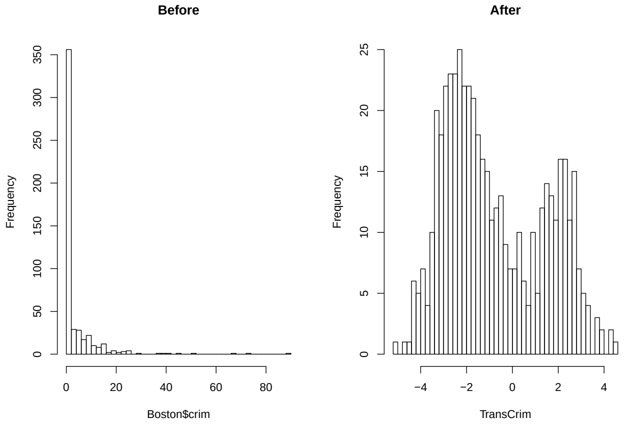
The value of λ is chosen using a maximum-likelihood argument. The R function `BoxCoxTrans` in the `caret` package can be used. (Subtracting 1 and dividing by λ makes the transformation approach a log as $\lambda \to 0$.)

Symmetrizing: R code

```
TransCrim = predict(BoxCoxTrans(Boston$crim),
          Boston$crim)
par(mfrow=c(1,2))
hist(Boston$crim, nclass=50, main = "Before")
hist(TransCrim, nclass=50, main = "After")
```

Note the use of the predict function to perform the actual transformation.

Before and after



Putting it together

The caret function PreProcess does both symmetrizing and scaling:

```
numericCols = unlist(lapply(Boston, is.numeric))
BostonCont = Boston[, numericCols]
BostonFactor =Boston[, !numericCols]
names(BostonFactor) = names(Boston)[!numericCols]
trans = preProcess(BostonCont, method = c("BoxCox",
                 "center", "scale"))
BostonTransScaledCont = predict(trans, BostonCont)
BostonTransScaled = data.frame(BostonTransScaledCont,
                  BostonFactor)
```

Data cleaning and Imputation

- Data cleaning means correcting any mistakes in the data, which will usually correspond to impossibly large or impossibly small values. These will be revealed by range checks and plots, for example all pairs of scatterplots, or qqplots.
- However, not all outliers (large or small values) are mistakes - they may reveal interesting patterns, for example the existence of two different groups of data in the data set.
- Imputation means guessing the values of missing values. Some of our methods (or at least the software implementations) require that there be no missing values in the input data set, although others (e.g.trees) are not unduly bothered by missing values.
- Data cleaning and imputation are a very important phase of any DM project and can be expected to absorb a big fraction of the project resources.

Data cleaning and Imputation

There are several methods for imputation:

- Complete case analysis: Delete any record that has missing values from the data set.
- Nearest neighbours: to impute variable x average the value x of the k closest data points with no missing values.
- Average method: Average the value of x for the non-missing values.
- Hot deck: pick a "similar" record at random and use its value of x.
- Predictive: Fit a model to the data with variable x as the target and use it to predict the value.
- Single imputation: Draw a value at random from the conditional distribution of x given the other variables (this will have to be modeled)
- Multiple imputation: Repeatedly draw values at random from the conditional distribution of x given the other variables (this will have to be modeled), creating new data sets. Make the predictions with these now complete datasets and average the predictions.

Points to note

- Complete case analysis: This is not recommended. Unless the missingness is independent of the data, biases can result.
- Nearest neighbours: We have to decide on the definition of "close points" and the value of k.
- Average method: Easy to implement but crude.
- Hot deck: Ditto.
- Predictive: Better but understates the uncertainty in the imputation process.
- Single imputation: Again better, respects the uncertainty, but just a single value.
- Multiple imputation: generally regarded as the best method (a sample is better than a single observation.)
- We will revist Multiple Imputation in Lecture 11.

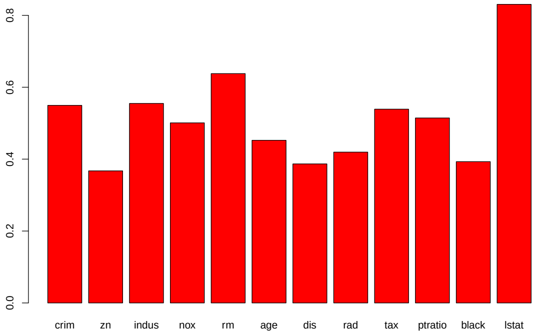
Feature selection

Features that are unrelated to the target are unlikely to improve the prediction, and will merely add to the computational burden. Thus, we can screen out variables that are unrelated, judged on the basis of correlations for continuous features, and anova tests for factors.

```
cors = cor(BostonScaled[,1:13])[1:12,13]
barplot(abs(cors), col="red")
cors = cor(BostonTransScaled[,1:13])[1:12,13]
barplot(abs(cors), col="red")
anova.fit = lm(medv~chas, data=BostonTransScaled)
anova(anova.fit)[1,5]
> anova(anova.fit)[1,5]
[1] 0.0002382315
```

(This is the p-value for the F-test that there is no difference in the response between the factor levels. Just use it as an index: keep features with small p-values (say less than 0.1)

Correlation plot



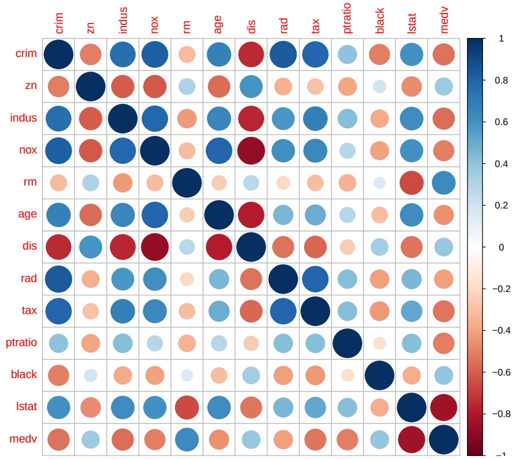
All seem to have some relationship to the target (recall that the best linear model included all the features)

Correlation between features

If two features are very highly correlated, it may not be necessary to include them both in the predictor (as in collinearity in linear regression). A correlation plot can be useful in identifying groups of highly correlated predictors.

```
library(corrplot)
correlations = cor(BostonTransScaled[,1:13])
corrplot(correlations)
```

Correlation plot



Correlation matrix

```
> round(correlations,2)
         crim    zn indus   nox    rm   age   dis   rad   tax ptratio black lstat  medv
crim     1.00 -0.52  0.75  0.81 -0.32  0.67 -0.74  0.84  0.79    0.41 -0.50  0.61 -0.55
zn      -0.52  1.00 -0.61 -0.61  0.31 -0.57  0.59 -0.35 -0.30   -0.39  0.19 -0.47  0.37
indus    0.75 -0.61  1.00  0.79 -0.42  0.66 -0.76  0.59  0.68    0.43 -0.38  0.63 -0.55
nox      0.81 -0.61  0.79  1.00 -0.31  0.80 -0.88  0.62  0.65    0.29 -0.40  0.61 -0.50
rm      -0.32  0.31 -0.42 -0.31  1.00 -0.25  0.27 -0.21 -0.30   -0.35  0.16 -0.65  0.64
age      0.67 -0.57  0.66  0.80 -0.25  1.00 -0.79  0.46  0.50    0.29 -0.31  0.62 -0.45
dis     -0.74  0.59 -0.76 -0.88  0.27 -0.79  1.00 -0.54 -0.58   -0.26  0.35 -0.54  0.39
rad      0.84 -0.35  0.59  0.62 -0.21  0.46 -0.54  1.00  0.80    0.42 -0.42  0.45 -0.42
tax      0.79 -0.30  0.68  0.65 -0.30  0.50 -0.58  0.80  1.00    0.42 -0.43  0.51 -0.54
ptratio  0.41 -0.39  0.43  0.29 -0.35  0.29 -0.26  0.42  0.42    1.00 -0.17  0.42 -0.51
black   -0.50  0.19 -0.38 -0.40  0.16 -0.31  0.35 -0.42 -0.43   -0.17  1.00 -0.37  0.39
lstat    0.61 -0.47  0.63  0.61 -0.65  0.62 -0.54  0.45  0.51    0.42 -0.37  1.00 -0.83
medv    -0.55  0.37 -0.55 -0.50  0.64 -0.45  0.39 -0.42 -0.54   -0.51  0.39 -0.83  1.00
```

See the R function `findCorrelation` in the `caret` package to see if any features should be removed.

Almost-zero-variance-features

If a feature has all values constant, it contributes nothing to the prediction. If almost all values are constant (e.g. a binary feature with very few 1's) then there may be problems with resampling, and the few 1's may have an undue influence on the predictor. Such features are best not included in the modeling. However, tree-based models are not affected by such features, as they won't be included in any split. Linear models will have problems if presented with zero-variance predictors if the software is not smart enough to exclude them (as `lm` does). The `caret` package has a function `nearZeroVar` that identifies almost-zero-variance features (see APM p 55)

```
> library(caret)
> nearZeroVar(BostonTransScaled)
integer(0)
# Returns column numbers of the
# near-zero-variance features
```

Dummy variables

You will recall that factors are handled in statistical modelling by turning them into dummy variables. Most of the modeling software does this automatically, but sometimes it is necessary to do this explicitly. There is a function (see APM p 56) `dummyVars` in the `caret` package that will do the job:

```
> dummy = dummyVars(~chas, data=BostonTransScaled)
> dummy.df = predict(dummy, newdata=BostonTransScaled)
> head(dummy.df)
  chas.0 chas.1
1      1      0
2      1      0
3      1      0
4      1      0
5      1      0
6      1      0
```

Feature engineering

Sometimes we want to make new variables our of old ones: a process known as feature engineering. Which new variables to make will be guided by subject matter knowledge. For example

1. Geographical coordinates could be turned into distances
2. Stock prices could be turned into log of the daily changes (returns)
3. Absolute numbers can be turned into rates
4. Several fertures can be combined into principal components (see later in the lectures on unsupervised learning)