



Preprocessing with Principal Components Analysis (PCA)

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Correlated predictors

```
library(caret); library(kernlab); data(spam)
inTrain <- createDataPartition(y=spam$type,
                               p=0.75, list=FALSE)

training <- spam[inTrain,]
testing <- spam[-inTrain,]

M <- abs(cor(training[, -58]))
diag(M) <- 0
which(M > 0.8, arr.ind=T)
```

	row	col
num415	34	32
num857	32	34

Correlated predictors

```
names(spam)[c(34,32)]
```

```
[1] "num415" "num857"
```

```
plot(spam[,34],spam[,32])
```

Basic PCA idea

- We might not need every predictor
- A weighted combination of predictors might be better
- We should pick this combination to capture the "most information" possible
- Benefits
 - Reduced number of predictors
 - Reduced noise (due to averaging)

We could rotate the plot

$$X = 0.71 \times \text{num415} + 0.71 \times \text{num857}$$

$$Y = 0.71 \times \text{num415} - 0.71 \times \text{num857}$$

```
X <- 0.71*training$num415 + 0.71*training$num857  
Y <- 0.71*training$num415 - 0.71*training$num857  
plot(X,Y)
```

Related problems

You have multivariate variables X_1, \dots, X_n so $X_1 = (X_{11}, \dots, X_{1m})$

- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is **statistical** and the second goal is **data compression**.

Related solutions - PCA/SVD

SVD

If X is a matrix with each variable in a column and each observation in a row then the SVD is a "matrix decomposition"

$$X = UDV^T$$

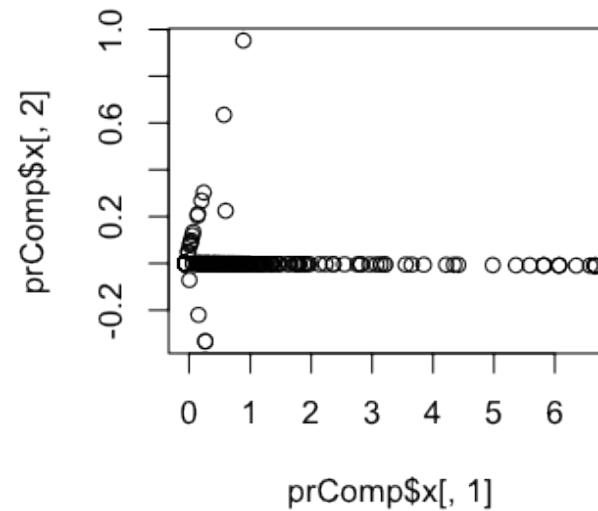
where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singular vectors) and D is a diagonal matrix (singular values).

PCA

The principal components are equal to the right singular values if you first scale (subtract the mean, divide by the standard deviation) the variables.

Principal components in R - prcomp

```
smallSpam <- spam[,c(34,32)]  
prComp <- prcomp(smallSpam)  
plot(prComp$x[,1],prComp$x[,2])
```



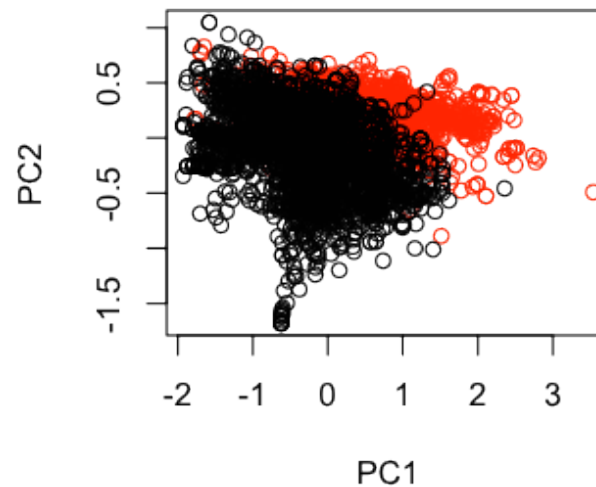
Principal components in R - prcomp

```
prComp$rotation
```

	PC1	PC2
num415	0.7081	0.7061
num857	0.7061	-0.7081

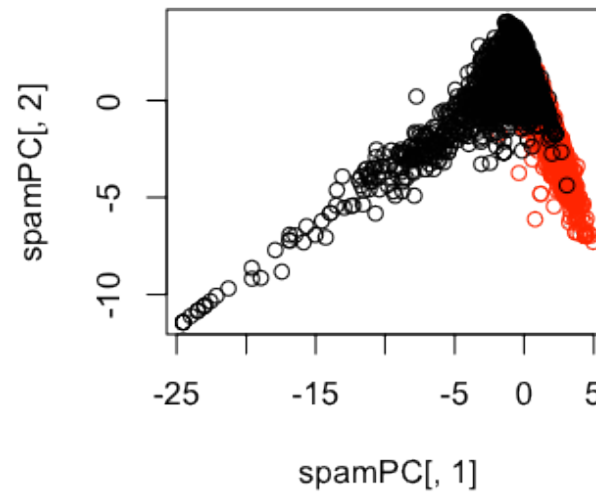
PCA on SPAM data

```
typeColor <- ((spam$type=="spam")*1 + 1)
prComp <- prcomp(log10(spam[, -58]+1))
plot(prComp$x[, 1], prComp$x[, 2], col=typeColor, xlab="PC1", ylab="PC2")
```



PCA with caret

```
preProc <- preProcess(log10(spam[, -58]+1), method="pca", pcaComp=2)  
spamPC <- predict(preProc, log10(spam[, -58]+1))  
plot(spamPC[, 1], spamPC[, 2], col=typeColor)
```



Preprocessing with PCA

```
preProc <- preProcess(log10(training[,-58]+1),method="pca",pcaComp=2)
trainPC <- predict(preProc,log10(training[,-58]+1))
modelFit <- train(training$type ~ .,method="glm",data=trainPC)
```

Preprocessing with PCA

```
testPC <- predict(preProc,log10(testing[,-58]+1))  
confusionMatrix(testing$type,predict(modelFit,testPC))
```

Confusion Matrix and Statistics

	Reference	
Prediction	nonspam	spam
nonspam	646	51
spam	64	389

Accuracy : 0.9

95% CI : (0.881, 0.917)

No Information Rate : 0.617

P-Value [Acc > NIR] : <2e-16

Kappa : 0.79

McNemar's Test P-Value : 0.263

Sensitivity : 0.910

Alternative (sets # of PCs)

```
modelFit <- train(training$type ~ .,method="glm",preProcess="pca",data=training)
confusionMatrix(testing$type,predict(modelFit,testing))
```

Confusion Matrix and Statistics

	Reference	
Prediction	nonspam	spam
nonspam	660	37
spam	54	399

Accuracy : 0.921

95% CI : (0.904, 0.936)

No Information Rate : 0.621

P-Value [Acc > NIR] : <2e-16

Kappa : 0.833

Mcnemar's Test P-Value : 0.0935

Sensitivity : 0.924

Final thoughts on PCs

- Most useful for linear-type models
- Can make it harder to interpret predictors
- Watch out for outliers!
 - Transform first (with logs/Box Cox)
 - Plot predictors to identify problems
- For more info see
 - Exploratory Data Analysis
 - [Elements of Statistical Learning](#)