The Principal Component Analysis

PCA reduces the data frame by orthogonally transforming the data into a set of principal components. The first principal component explains the most amount of the variation in the data in a single component while the second component explains the second most amount of the variation and so on. By choosing the top k principal components that explain say 80-90% of the variation, the other components can be dropped since they do not significantly benefit the model. This procedure retains some of the latent information in the principal components which can help to build better models.

> suppressPackageStartupMessages(library(DAAG))

> head(ais)

rcc wcc hc hg ferr bmi ssf pcBfat lbm ht wt sex sport

1 3.96 7.5 37.5 12.3 60 20.56 109.1 19.75 63.32 195.9 78.9 f B\_Ball

2 4.41 8.3 38.2 12.7 68 20.67 102.8 21.30 58.55 189.7 74.4 f B\_Ball

3 4.14 5.0 36.4 11.6 21 21.86 104.6 19.88 55.36 177.8 69.1 f B\_Ball

4 4.11 5.3 37.3 12.6 69 21.88 126.4 23.66 57.18 185.0 74.9 f B\_Ball

5 4.45 6.8 41.5 14.0 29 18.96 80.3 17.64 53.20 184.6 64.6 f B\_Ball

6 4.10 4.4 37.4 12.5 42 21.04 75.2 15.58 53.77 174.0 63.7 f B\_Ball

# standardize

minmax <- function(x) (x - min(x))/(max(x) - min(x))

x\_train <- apply(ais[,1:11], 2, minmax)

# PCA

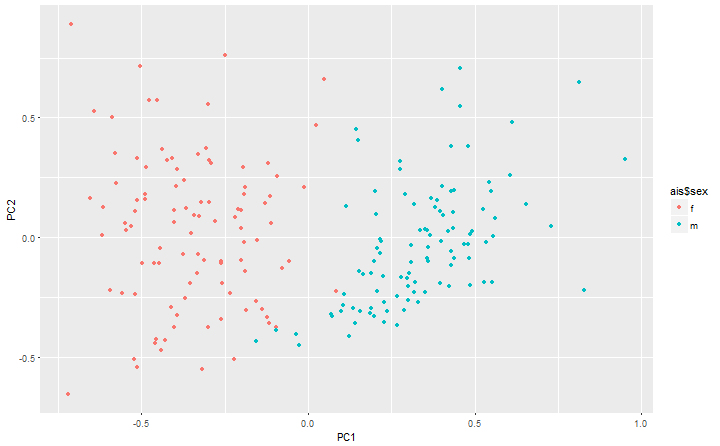
pca <- prcomp(x\_train)

# plot cumulative plot

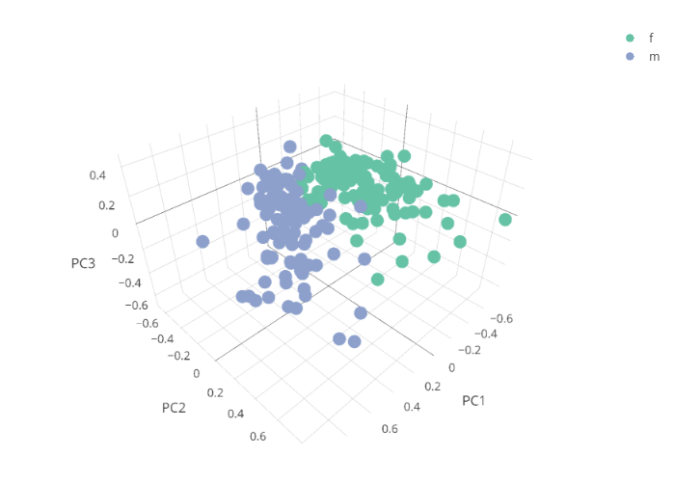
> qplot(x = 1:11, y = cumsum(pca$sdev)/sum(pca$sdev), geom = "line")



> gplot (as.data.frame (pca$x), aes(x=PC1, y=PC2, col=ais$sex)) + geom point()



> pca\_plotly <- plot\_ly(as.data.frame(pca$x), x = ~PC1, y = ~PC2, z = ~PC3, color = ~ais$sex) %>% add\_markers()



The autoencoder

# autoencoder in keras

> suppressPackageStartupMessages(library(keras))

# set training data

> x\_train <- as.matrix(x\_train)

# set model

> model <- keras\_model\_sequential()

> model %>%

+ layer\_dense(units = 6, activation = "tanh", input\_shape = ncol(x\_train)) %>%

+ layer\_dense(units = 2, activation = "tanh", name = "bottleneck") %>%

+ layer\_dense(units = 6, activation = "tanh") %>%

+ layer\_dense(units = ncol(x\_train))

# view model layers

> summary(model)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

===========================================================================

dense\_70 (Dense) (None, 6) 72

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bottleneck (Dense) (None, 2) 14

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_71 (Dense) (None, 6) 18

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_72 (Dense) (None, 11) 77

===========================================================================

Total params: 181

Trainable params: 181

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# compile model

> model %>% compile(loss = "mean\_squared\_error", optimizer = "adam")

# fit model

> model %>% fit(x = x\_train, y = x\_train, epochs = 2000,verbose = 0)

# evaluate the performance of the model

mse.ae2 <- evaluate(model, x\_train, x\_train)

mse.ae2

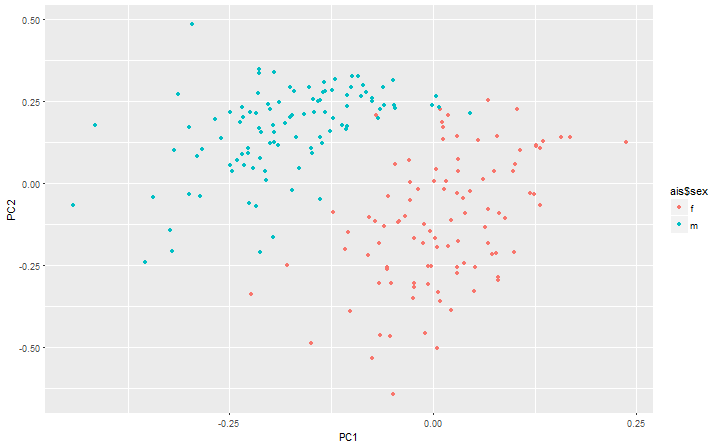
loss 0.009725631

# extract the bottleneck layer

intermediate\_layer\_model <- keras\_model(inputs = model$input, outputs = get\_layer(model, "bottleneck")$output)

intermediate\_output <- predict(intermediate\_layer\_model, x\_train)

> ggplot(data.frame(PC1 = intermediate\_output[,1], PC2 =intermediate\_output[,2]), aes(x = PC1, y = PC2, col = ais$sex)) + geom\_point()



The autoencoder is still separating the males from the females, however it picks up on structure in the data that PCA does not. Though, this is a small data set with only 11 variables the autoencoder does not pick up on too much more than the PCA. Highly complex data with perhaps thousands of dimensions the autoencoder has a better chance of unpacking the structure and storing it in the hidden nodes by finding hidden features. In contrast to PCA the autoencoder has all the information from the original data compressed into the reduced layer. To view the data in 3 dimensions the model will need to be fited again with the bottleneck layer with 3 nodes.

model3 <- keras\_model\_sequential()

< model3 %>% layer\_dense(units = 6, activation = "tanh", input\_shape = ncol(x\_train)) %>%layer\_dense(units = 3, activation = "tanh", name = "bottleneck") %>%layer\_dense(units = 6, activation = "tanh")

%>%layer\_dense(units = ncol(x\_train))

# summary of model

< summary(model3)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

===========================================================================

dense\_73 (Dense) (None, 6) 72

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bottleneck (Dense) (None, 3) 21

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_74 (Dense) (None, 6) 24

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_75 (Dense) (None, 11) 77

===========================================================================

Total params: 194

Trainable params: 194

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# compile model

< model3 %>% compile(loss = "mean\_squared\_error", optimizer = "adam")

# fit model

< model3 %>% fit( x = x\_train, y = x\_train, epochs = 2000, verbose = 0)

# evaluate the model

< evaluate(model3, x\_train, x\_train)

loss 0.006243833

# exgtract the bottleneck layer

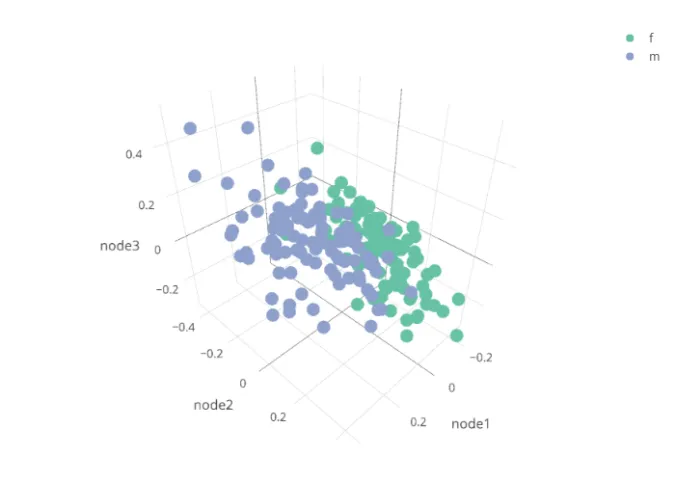
intermediate\_layer\_model <- keras\_model(inputs = model3$input, outputs = get\_layer(model3, "bottleneck")$output)

intermediate\_output <- predict(intermediate\_layer\_model, x\_train)

# plot the reduced data set

aedf3 <- data.frame(node1 = intermediate\_output[,1], node2 = intermediate\_output[,2], node3 = intermediate\_output[,3])

ae\_plotly <- plot\_ly(aedf3, x = ~node1, y = ~node2, z = ~node3, color = ~ais$sex) %>% add\_markers()



Conclusively, one can see that there is a slight difference between the autoencoder and PCA plots and perhaps the autoencoder does slightly better at differentiating between male and female athletes.