portfolio_3

Henry Bourne

2023-02-06

Kernel Principal Component Analysis

First we will generate our training dataset that we will use for this task and visualize it:

```
library(plotly)
```

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

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(Rfast)
```

```
## Loading required package: Rcpp
```

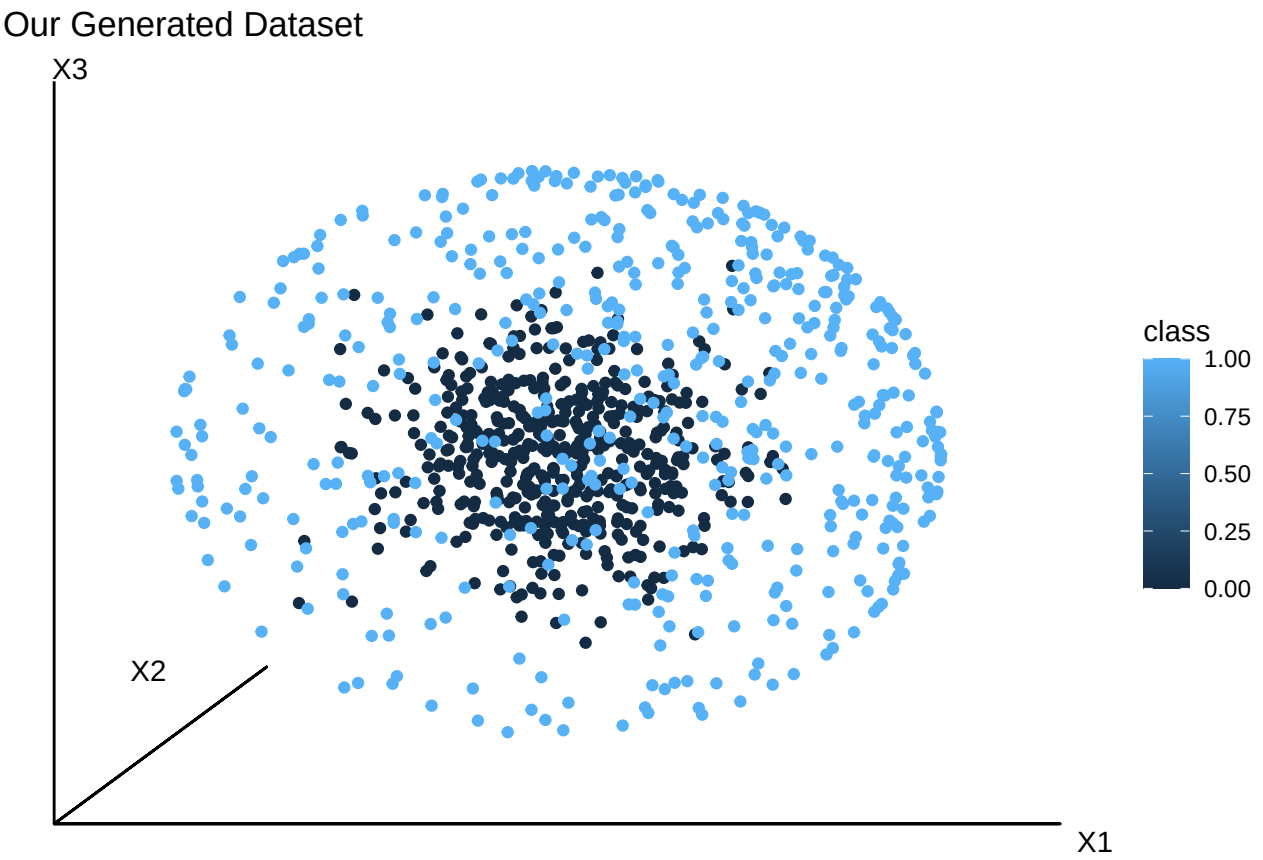
```
## Loading required package: RcppZiggurat
```

```
n <- 1000
d <- 3
C0 <- rmvnorm(n/2, rep(0,d), diag(rep(0.06, d)))
C1 <- rvmf(n/2, mu = c(1, 1, 1) / sqrt(10), k = 1)
D <- data.frame(rbind(C0, C1), "class"= c(rep(0,n/2), rep(1,n/2) ) )

library("gg3D")
ggplot(D, aes(x=X1, y=X2, z=X3, color=class)) +
  theme_void() +
  axes_3D(theta= 0, phi =0) +
  stat_3D(theta=0, phi=0) +
  labs_3D(theta=-13, phi=10,  labs=c("X1", "X2", "X3"),
            angle=c(0,0,0),
            hjust=c(0,2,2),
            vjust=c(2,2,-2)) +
  ggtitle("Our Generated Dataset")
```

```
## Warning: The following aesthetics were dropped during statistical transformation: z,
```

```
## colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation: z,
## colour
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



As you can see this dataset is clearly not linearly separable, however, it is indeed separable. We will use this dataset in our task to perform classification using PCA and KPCA and perform an analysis on classification performance. Let's now generate the test dataset which we will use later to evaluate the performance of our trained classifiers:

```
n.test <- 200
C0.test <- rmvnorm(n.test/2, rep(0,d), diag(rep(0.06, d)))
C1.test <- rvmf(n.test/2, mu = c(1, 1, 1) / sqrt(10), k = 1)
Test <- data.frame(rbind(C0.test, C1.test), "class"= c(rep(0,n.test/2), rep(1,n.test/2) ) )
```

PCA vs KPCA

In this subsection we will perform binary classification using data dimensionality reduction performed by PCA and KPCA. We will then compare their performance.

Let's now move onto performing our principal component analyses. First lets carry out PCA on the data

matrix, note that we won't scale or center the data as its already centered and scaled. By not scaling we get the bonus of preserving variance in our transformation:

```
pc <- prcomp(~ . -class, D, scale. = FALSE, retx=TRUE)
pc
```

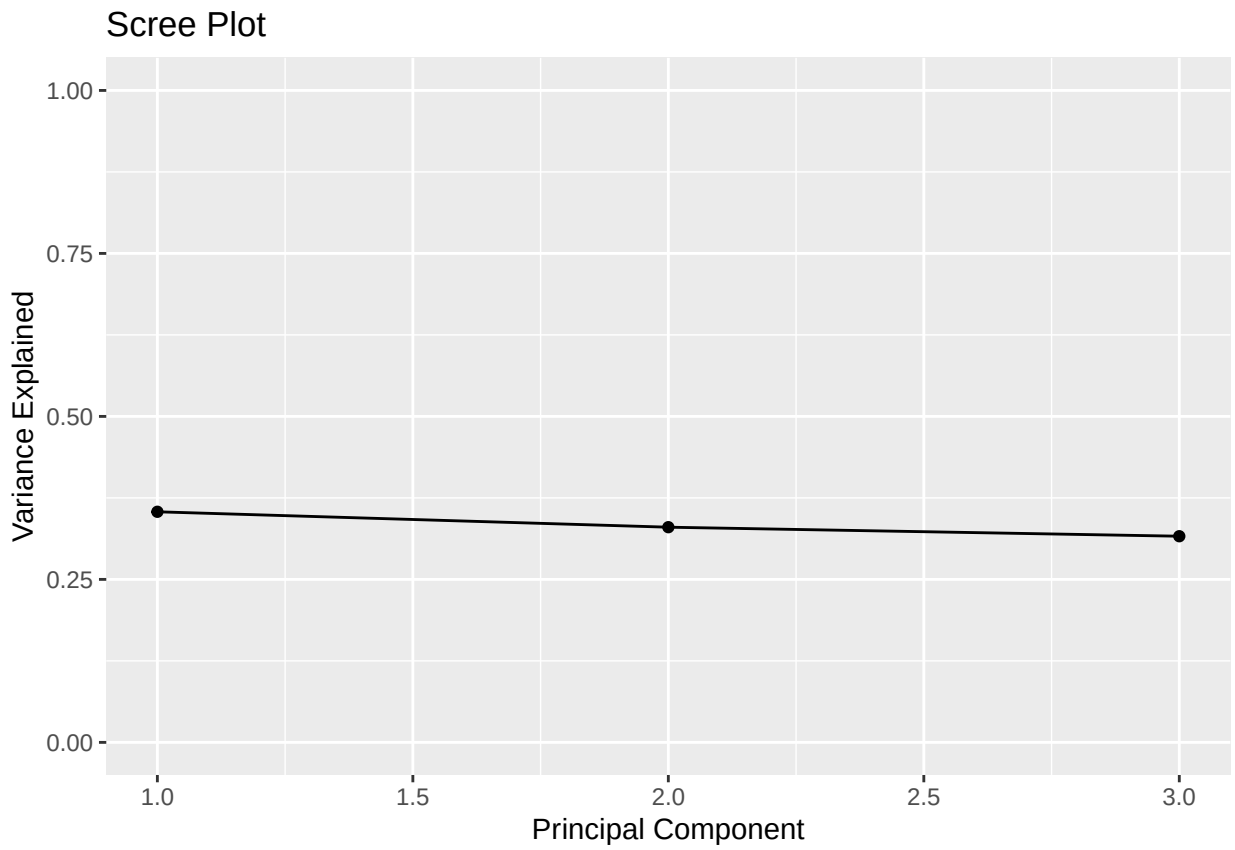
```
## Standard deviations (1, .., p=3):
## [1] 0.4462212 0.4310422 0.4219142
##
## Rotation (n x k) = (3 x 3):
##           PC1        PC2        PC3
## X1  0.8161587 -0.3395218  0.4675574
## X2 -0.5469416 -0.7149342  0.4355731
## X3  0.1863863 -0.6112234 -0.7691984
```

Let's produce a scree plot and select the number of principal components we want to use:

```
var_prcnt = pc$sdev^2 / sum(pc$sdev^2)

qplot(c(1:3), var_prcnt) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```



We see that the variance accounted for by each principal component is roughly the same, which makes sense

as looking at the plot above the variance is roughly the same in every direction. Hence, we will keep all three principal components for the rest of the analysis.

Now let's perform KPCA, using the radial basis kernel function and again we will not scale or center the data:

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
kpc.0 <- kpca(~ . -class, D, kernel="rbfdot")
head(pcv(kpc.0))
```

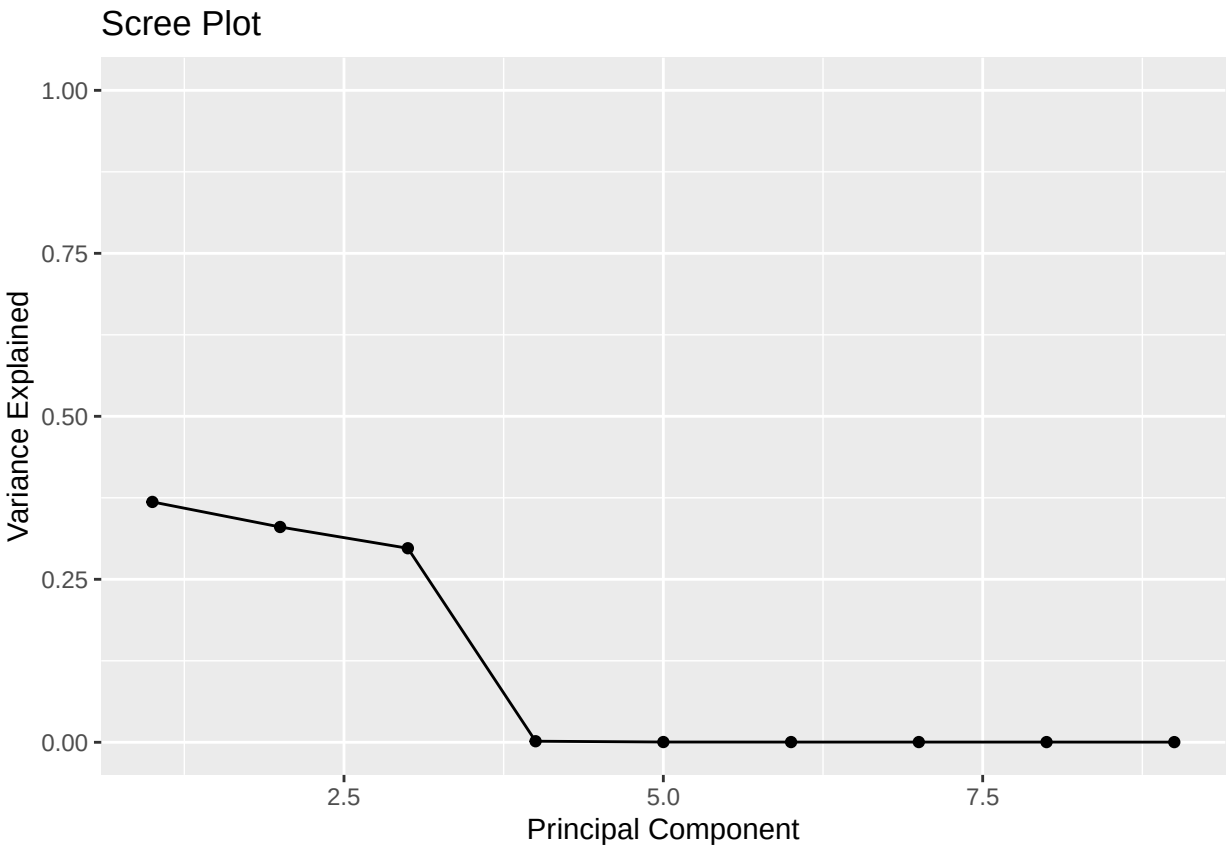
```
##              [,1]        [,2]         [,3]       [,4]       [,5]        [,6]
## [1,] -0.209842980 -0.13741256 -0.056209741 -0.2355042 -1.0403127 -0.19975414
## [2,] -0.037509561 -0.10448044  0.105741954 -0.7774402 -0.1789189 -0.09198307
## [3,] -0.001884043  0.10321832 -0.074323084 -0.7950585 -0.0753221 -0.05227167
## [4,] -0.109209475 -0.06202994  0.034431835 -0.7753013 -0.4790691 -0.07520606
## [5,] -0.002116527  0.06162952  0.008372452 -0.9048919 -0.1863247  0.04347401
## [6,] -0.181490690 -0.15776309  0.042262657 -0.3452700 -0.9174523 -0.65344519
##             [,7]        [,8]        [,9]
## [1,]  0.41568091 -0.68680219 -0.09108769
## [2,]  0.06111812  0.17505233  0.31029878
## [3,]  0.13559013  0.15615091  0.15049626
## [4,] -0.08038223 -0.09876851  0.20564943
## [5,] -0.02171514 -0.06444817  0.06295063
## [6,]  0.17617682 -0.11779542  0.08350326
```

```
# We only print first 6 rows
#TODO: comment on dimensionality of component vectors
```

Let's again create a scree plot in order to help us select the number of principal components we would like to keep:

```
var_prcnt = eig(kpc.0)^2 / sum(eig(kpc.0)^2)

qplot(c(1:length(eig(kpc.0))), var_prcnt) +
  geom_line() +
  xlab("Principal Component") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```



From the scree plot it's very clear that the first three principal components contain nearly all the variance, hence, we will use only the first three components. Let's update our KPCA to reflect this choice:

```
kpc.0 <- kpca(~ . -class, data=D, kernel="rbfdot", features=3)
```

Let's now train a logistic model on the transformed data from our PCA

```
D.rt <- data.frame(pc$x, "class"=D[,4])
model.pc <- glm(class~ ., family=binomial(link='logit'), data=D.rt)
model.pc
```

```
##
## Call:  glm(formula = class ~ ., family = binomial(link = "logit"), data = D.rt)
##
## Coefficients:
## (Intercept)          PC1          PC2          PC3
##    0.004476     0.648571    -1.948313     0.171524
##
## Degrees of Freedom: 999 Total (i.e. Null);  996 Residual
## Null Deviance:       1386
## Residual Deviance: 1226  AIC: 1234
```

And now on our transformed data from our KPCA:

```
D.rt <- data.frame(rotated(kpc.0), "class"=D[,4])
model.kpc.0 <- glm(class~ ., family=binomial(link='logit'), data=D.rt)
model.kpc.0
```

```
##
```

```
## Call:  glm(formula = class ~ ., family = binomial(link = "logit"), data = D.rt)
##
## Coefficients:
## (Intercept)           X1           X2           X3
##    0.008712     0.057442    -0.157049     0.015202
##
## Degrees of Freedom: 999 Total (i.e. Null);  996 Residual
## Null Deviance:       1386
## Residual Deviance: 1208  AIC: 1216
```

Let's now test our models' performance on the testing data:

```
# Let's first obtain our predictions for the test data and calculate the accuracy for PCA
Test.rt <- data.frame(as.matrix(Test[,-4]) %*% pc$rotation, "class" = Test[,4])
preds.pc <- predict(model.pc, Test.rt, type="response")
preds.pc <- ifelse(preds.pc > 0.5,1,0)
acc.pc <- 1-mean(preds.pc != Test[,4])

# Let's now obtain our predictions for the test data and calculate the accuracy for KPCA
Test.rt <- predict(kpc.0,Test)
Test.rt <- data.frame("X1" = Test.rt[,1], "X2" = Test.rt[,2], "X3" = Test.rt[,3], "class" = Test[,4])
preds.kpc.0 <- predict(model.kpc.0, Test.rt, type="response")
preds.kpc.0 <- ifelse(preds.kpc.0 > 0.5,1,0)
acc.kpc.0 <- 1-mean(preds.kpc.0 != Test[,4])


# Let's now print the accuracies we obtained
acc.pc # accuracy using PCA
```

```
## [1] 0.61
```

```
acc.kpc.0 # accuracy using KPCA
```

```
## [1] 0.705
```

We note that the