Kernel PCA

Mathias Bourel

31 de May de 2021

Resumen

Resumen de la técnica Kernel PCA. Basado en los materiales [2] y [1]

Índice

Describa con precisión la técnica Kernel PCA y cuales son las ventajas respecto del Análisis de Componentes Principales (PCA) clásico. Se prestará particular atención al *"kernel trick"*. En los distintos materiales que busque, encontrar un conjunto de datos, por ejemplo el de los tres círculos concéntricos, en donde es más adecuado aplicar Kernel PCA que PCA. Implementar y justificar.

1. Introducción - Círculos concéntricos

La figura 1 muestra un conjunto de puntos dispuestos en forma de circunferencias concéntricos de radios 4, 9 y 10 (con ruido aleatorio superpuesto en las coordenadas X e Y). Un problema en el que se puede trabajar es el de la agrupación de los puntos de acuerdo a qué circunferencia pertenezcan. La figura 3 ilustra esta clasificación realizada manualmente, donde los puntos aparecen coloreados según su radio.

```
R = c(4, 9, 16)
theta = seq(from = 0, to = 2*pi, length.out = 100)

puntos  <- expand.grid(R, theta)

set.seed(852147)
ruidoX  <- rnorm(n = length(theta), mean = 0, sd = 0.4)
ruidoY  <- rnorm(n = length(theta), mean = 0, sd = 0.4)

puntos  <- cbind(puntos,
                 puntos[,1]*cos(puntos[,2]) + ruidoX,
                 puntos[,1]*sin(puntos[,2]) + ruidoY
                 )
puntos  <- cbind(puntos,
                 sqrt(puntos[,3]^2 + puntos[,4]^2)
                 )

puntos <- data.frame(puntos)
puntos[,1] <- as.factor(puntos[,1])
colnames(puntos) = c("R", "Theta", "X", "Y", "Z")

circCon <- data.frame(puntos$R, puntos$X, puntos$Y)

ggplot(puntos, aes(X, Y)) + geom_point() + coord_fixed()
```

Observemos que para este conjunto de punto no es posible obtener una separación satisfactoria con PCA que permita hacer una clasificación de los mismos en regiones diferentes. En la figura 2 vemos el resultado de aplicar PCA. La representación en el plano factorial basta para explicar el 100 % de la variabilidad de los datos, y vemos que el mapeo obtenido es esencialmente el mismo que el original. Mediante este método no se genera ninguna información adicional que permita identificar cada circunferencia.

```
# PCA
puntosPCA <- as.matrix(scale(x = puntos[3:4], center = TRUE, scale = TRUE))
acp1 <- FactoMineR::PCA(puntosPCA, graph = FALSE)
fviz_pca_ind(acp1, label="none") + theme_gray()
```

Hagamos una clasificación manual de los datos (Figura 3).

```
ggplot(puntos, aes(X, Y, colour = R)) + geom_point() + coord_fixed()
```

Una primera aproximación para generar información adicional se muestra en la figura 4, donde se ha agregado una tercera dimensión, representada en el eje (Oz). Las coordenadas de los puntos sobre este eje será dada por $z = \sqrt{x^2 + y^2}$ y representa el radio de cada circunferencia (a menos del ruido agregado).

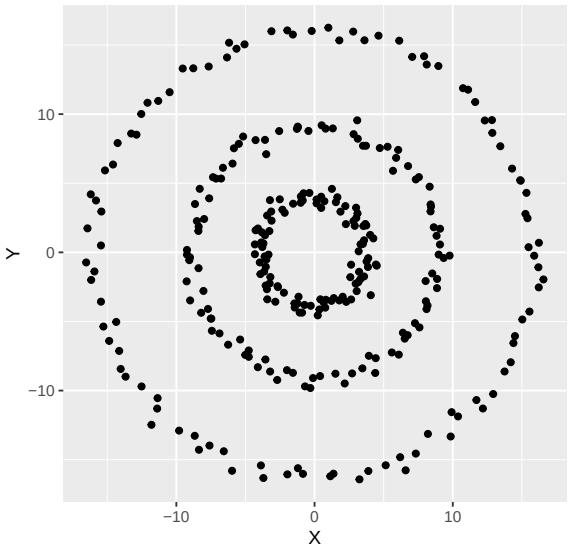

Figura 1: Ejemplo de círculos concéntricos - radios 4, 9 y 16.

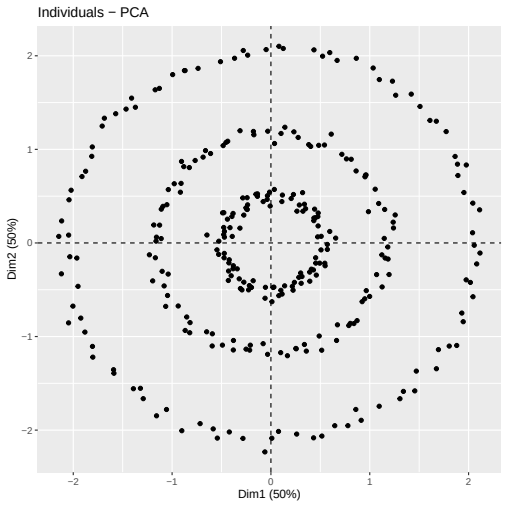

Figura 2: Representación en el plano (únicos ejes factoriales) del ejemplo de círculos concéntricos, obtenida por PCA.

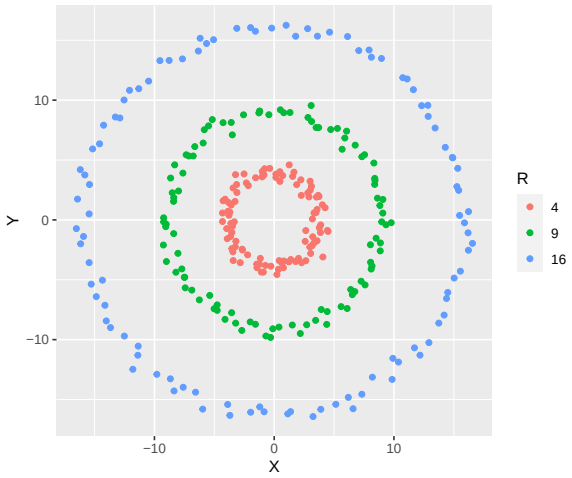

Figura 3: Clasificación manual de círculos concéntricos.

```
scatter3D(puntos$X, puntos$Y, puntos$Z, bty = "b2", phi=25)
```

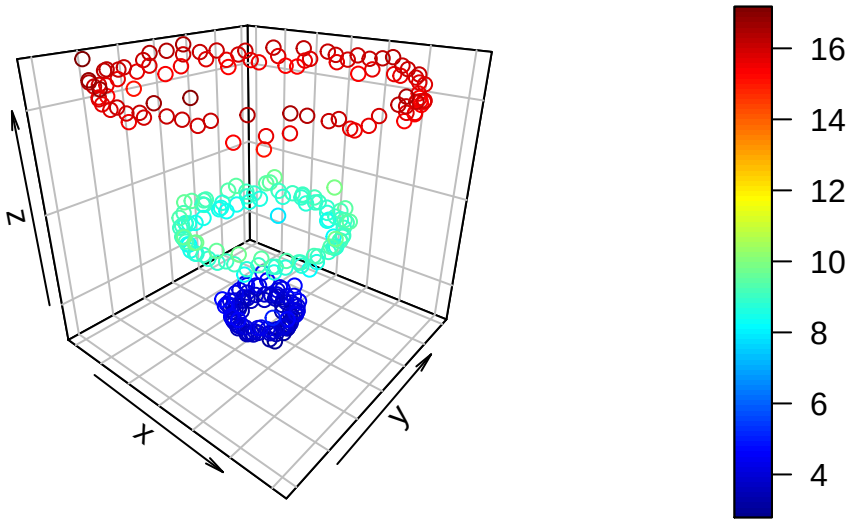

Figura 4: Mapeo de los puntos en 3D, con coordenada $Z = R^2$.

Observamos que podemos visualizar posibles separaciones lineales entre los conjuntos de puntos en un espacio de una dimensión más grande que el original y podriamos tratar de aplicar PCA en este nuevo espacio para ver si logramos "separar" los puntos originales.

2. Kernel trick

El truco del nucleo (kernel trick) se usa en problemas donde la solución depende del producto escalar entre dos vectores. La idea consiste, mediante una función ϕ en llevar puntos originales a un espacio de dimensión grande y de remplazar el producto escalar $\phi(x)\phi(y)$ por una función K de x,y mediante $K(x,y) = \phi(x)\dot{\phi}(y)$. El teorema de Mercer nos asegura que si la función K cumple que

$$\sum_{i,j} K(x_i,x_j)w_i w_j \geq 0$$

para todos $x_1,\ldots,X_N \in \mathbb{R}^d$ y para todos reales $c_1,\ldots,c_N$) entonces existe una función ϕ con codominio un espacio de Hilbert de gran dimensión (infinito) tal que $K(x,y) = \phi(x)\dot{\phi}(y)$.

Los puntos originales $x_1,x_2,\ldots,x_N \in \mathbb{R}^d$ a un nuevo espacio de dimensi'on mucho m'as grande que llamaremos *"espacio de características"* a través de una *función de mapeo de características* ϕ.

$$K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}, \quad K(x_1,x_2) = \phi(x_1)'\phi(x_2)$$

siendo K semidefinida positiva.

Algunas funciones de kernel de uso común son:

- **Gaussiano:** $K(x_1,x_2) = \exp\left(-\frac{||x_1-x_2||^2}{2\sigma^2}\right)$
- **Polinómico:** $K(x_1,x_2) = (c + x_1 \cdot x_2)^p$
- **Tangente Hiperbólico** $K(x_1,x_2) = \tanh(x_1 \cdot x_2 + d)$

La cantidad de parámetros para configurar las funciones de kernel es variable. Por ejemplo en el caso del kernel gaussiano, basta con indicar el parámetro σ, mientras que para configurar el kernel polinómico es necesario especificar los valores de c y p.

El kernel gaussiano puede pensarse como un estimador de la similaridad entre los puntos x_1 y x_2: es una función semidefinida positiva, vale $1 \iff x_1 = x_2$ y está cercano a cero cuando los vectores x_1 y x_2 son cada vez más distantes.

3. Kernel PCA

- La idea consiste en realizar un PCA en un espacio de diemnsión más grande, usando el kernel trick
- En vez de considerar los puntos $x_1,\ldots,x_n$ en el espacio original $\mathbb{R}^d$ los "mapeamos" en el espacio de caracteristicas obteniendo un nuevo conjunto de puntos $\phi(x_1),\ldots,\phi(x_n)$
- Calculamos las componentes principales en este nuevo espacio
- El resultado es no lineal en el espacio original.

3.1. Estudio en el espacio de datos mapeados

Supongamos que tenemos un conjunto de N puntos centrados $x_1,\ldots,x_N$ en el espacio inicial (de dimensión baja d). Recordar que si Σ es la matriz de varianzas covarianzas entonces

$$\Sigma = \frac{1}{N}\sum_{i=1}^{N} x_i x_i^{'} \tag{1}$$

Con el objetivo de aplicar PCA en el nuevo espacio de características, se calcula la *matriz de covarianzas de los datos mapeados* en el espacio de características de dimensión D. Consideramos entonces la matriz C definida por:

$$C = \frac{1}{N}\sum_{i=1}^{N} \phi(x_i)\phi(x_i)^{'} \tag{2}$$

Asumiremos en un primer momento que los datos mapeados están centrados, esto es que $\frac{1}{N}\sum_{i=1}^{N}\phi(x_i) = 0$. Más adelante veremos como generalizamos esta condición para así evitarla.

Las componentes principales en el espacio de características se calculan a partir de los valores y vectores propios de C.

Teorema: Los vectores propios v de C se pueden escribir como combinación lineal de los datos mapeados es decir:

$$v = \sum_{i=1}^{N} \alpha_i \ \phi(x_i) \tag{3}$$

Demostración: Para probarlo, necesitamos del resultado técnico siguiente. Sean x y v dos vectores en $\mathbb{R}^M$, entonces:

$$x \ x' \ v = (x \cdot v)\, x \tag{4}$$

donde los dos productos de la izquierda son productos matriciales, el primer producto de la derecha es un producto escalar y el segundo es matricial. Esto se cumple, ya que

$$(x \ x') \ v = \begin{pmatrix} x_1x_1 & x_1x_2 & \dots & x_1x_M \\ x_2x_1 & x_2x_2 & \dots & x_2x_M \\ \vdots & \vdots & \ddots & \vdots \\ x_Mx_1 & x_Mx_2 & \dots & x_Mx_M \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} = \begin{pmatrix} x_1x_1v_1 + x_1x_2v_2 + \cdots + x_1x_Mv_M \\ x_2x_1v_1 + x_2x_2v_2 + \cdots + x_2x_Mv_M \\ \vdots \\ x_Mx_1v_1 + x_Mx_2v_2 + \cdots + x_Mx_Mv_M \end{pmatrix} =$$

$$= \begin{pmatrix} (x_1v_1 + x_2v_2 + \cdots + x_Mv_M) \ x_1 \\ (x_1v_1 + x_2v_2 + \cdots + x_Mv_M) \ x_2 \\ \cdots \\ (x_1v_1 + x_2v_2 + \cdots + x_Mv_M) \ x_M \end{pmatrix} = \left(x_1v_1 + x_2v_2 + \cdots + x_Mv_M\right) \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} = (x \cdot v) \ x$$

Consideremos ahora v un vector propio de C asociado al valor propio λ, es decir $Cv = \lambda v$. Entonces por la expresión dada en la ecuación (2) y aplicando el resultado anterior (4) obtenemos que :

$$v = \frac{1}{\lambda}\left(\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)'\right)v = \frac{1}{N\lambda}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)'v = \sum_{i=1}^{N}\underbrace{\frac{(\phi(x_i)\cdot v)}{N\lambda}}_{\alpha_i} \ \phi(x_i)$$

Por lo tanto cualquier solución del problema de vectores propios de C (con $\lambda \neq 0$) se encuentra en el subespacio generado por $\phi(x_1), \dots, \phi(x_n)$ ya que $v = \sum limits_{i=1}^{N} \alpha_i \phi(x_i)$. Consecuentemente, encontrar los vectores propios v de C equivale en encontrar los N coeficientes α_i de v en el subespacio generado por $\phi(x_1), \dots, \phi(x_n)$.

Con los expresiones anteriores, volviendo a $Cv = \lambda v$, si sustituimps C a partir de (2) y si v_j es vector propio de C asociado a λ_j, entonces, a partir de (3), obtenemos:

$$\left(\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)'\right)\left(\sum_{l=1}^{N}\alpha_{jl} \ \phi(x_l)\right) = \lambda_j\left(\sum_{l=1}^{N}\alpha_{jl} \ \phi(x_l)\right)$$

A partir del resultado (4), podemos expresar lo anterior como:

$$\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\left(\sum_{l=1}^{N}\alpha_{jl} \ \phi(x_i)'\phi(x_l)\right) = \lambda_j\sum_{l=1}^{N}\alpha_{jl} \ \phi(x_l)$$

Si sustituimos $\phi'(x_i)\phi(x_l)$ en la expresión anterior por $K(x_i, x_l)$, obtenemos:

$$\frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\sum_{l=1}^{N}\alpha_{jl}\ K(x_i,x_l)=\lambda_j\sum_{l=1}^{N}\alpha_{jl}\ \phi(x_l)$$

Multiplicamos a la izquierda por $\phi(x_k)'$:

$$\frac{1}{N}\sum_{i=1}^{N}\phi(x_k)'\phi(x_i)\sum_{l=1}^{N}\alpha_{jl}\ K(x_i,x_l)=\lambda_j\sum_{l=1}^{N}\alpha_{jl}\ \phi'(x_k)\phi(x_l)$$

y sustituyendo $\phi(x_k)'\phi(x_i)$ por $K(x_k,x_i)$ y $\phi(x_k)'\phi(x_l)$ por $K(x_k,x_l)$ se tiene:

$$\frac{1}{N}\sum_{i=1}^{N}K(x_k,x_i)\sum_{l=1}^{N}\alpha_{jl}\ K(x_i,x_l)=\lambda_j\sum_{l=1}^{N}\alpha_{jl}\ K(x_k,x_l)$$

Matricialmente, lo anterior se puede expresar como:

$$K^2\alpha_{\mathbf{j}}=N\lambda_j K\alpha_{\mathbf{j}}$$

siendo $\alpha_{\mathbf{j}}$ el vector que contiene los coeficientes α_{jl}, con $1\leq l\leq N$. Podemos simplificar de ambas lados por K obteniendo:

$$K\alpha_{\mathbf{j}}=N\lambda_j\alpha_{\mathbf{j}}$$

Nuestro problema se reduce entonces en encontrar los vectores propios $\alpha_{\mathbf{j}}$ de la matriz de kernel K, con valores propios asociados iguales a $N\lambda_j$. Imponemos que los vectores v_j sean unitarios, lo cual tiene su implicancia sobre $\alpha_{\mathbf{j}}$ ya que:

$$v_j'\ v_j=1\Rightarrow\sum_{k,l=1}^{N}\alpha_{jl}\ \alpha_{jk}\ \phi(x_l)'\ \phi(x_k)=1\Rightarrow\alpha_{\mathbf{j}}'\ K\alpha_{\mathbf{j}}=1$$

Volviendo a usar que $K\alpha_{\mathbf{j}}=N\lambda_j\alpha_{\mathbf{j}}$, multiplicando por $\alpha_{\mathbf{j}}{}'$ y usando esta condición de normalización tenemos entonces que:

$$\lambda_j\ N\ \alpha_{\mathbf{j}}{}'\alpha_{\mathbf{j}}=1,\forall j$$

Para un nuevo punto x, su proyección sobre las componentes principales será dada por:

$$\phi(x)'\ v_j=\sum_{i=1}^{N}\alpha_{ji}\ \phi(x)'\ \phi(x_i)=\sum_{i=1}^{N}\alpha_{ji}\ K(x,x_i)$$

3.2. Observaciones:

- **Kernel trick.** La función de mapeo de características ϕ no es necesaria conocerla explicitamente, ni siquiera conocer D. En las cuentas, solo aparecen productos internos $\phi(x_i)'\phi(x_j)$ que por el teorema de Mercer se sustituyen por $K(x_i,x_j)$. Esto es lo que se llama el *kernel trick*.

- **Vectores de características no centrados.** En general los puntos $\phi(x_1), \ldots, \phi(x_n)$ no están centrados. Lo que se hace es centrarlos con

$$\tilde{\phi}(x_i) = \phi(x_i) - \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)$$

y el n'ucleo correspondiente es

$$\tilde{K}(x_i,x_j) = \tilde{\phi}(x_i)'\tilde{\phi}(x_j) = \left(\phi(x_i) - \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\right)' \left(\phi(x_j) - \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\right)$$

$$= K(x_i,x_j) - \frac{1}{N}\sum_{k=1}^{N}K(x_i,x_k) - \frac{1}{N}\sum_{k=1}^{N}K(x_j,x_k) + \frac{1}{N^2}\sum_{l,k=1}^{N}K(x_l,x_k)$$

La forma matricial normalizada de K resulta ser:

$$\tilde{K} = K - 2\ \mathbf{1}_{1/N}\ K + \mathbf{1}_{1/N}\ K\ \mathbf{1}_{1/N}$$

donde $\mathbf{1}_{1/N}$ es una matriz con todos sus elementos iguales a $\frac{1}{N}$. Luego se resuelve el problema de valores propios con la matriz normalizada

- El dato original $x \in \mathbb{R}^d$ se transformará entonces sobre el plano factorial en $y \in \mathbb{R}^{d'}$ con $d' \leq d$ y la coordenada $j-$esima de y será:

$$y_j = \sum_{i=1}^{N}\alpha_{ij}\ \tilde{K}(x,x_i) \qquad j = 1,\ldots,d$$

- **Resumen de kernel PCA:**

1. Se elige una función de kernel
2. Se construye la matriz de kernel normalizada de los datos

$$\tilde{K} = K - 2\ \mathbf{1}_{1/N}\ K + \mathbf{1}_{1/N}\ K\ \mathbf{1}_{1/N}$$

3. Se resuelve el problema de valores y vectores propios de $\tilde{K}$:

$$\tilde{K}\alpha_i = \lambda_i\alpha_i$$

4. Cada punto x se puede representar como un punto y donde la $j-$esima coordenada de y es:

$$y_j = \sum_{i=1}^{N}\alpha_{ij}\ \tilde{K}(x,x_i) \qquad j = 1,\ldots,d$$

4. Aplicación de kernel PCA al ejemplo de los círculos concéntricos

Sobre este conjunto de puntos se aplica kernel PCA con la función `kpca()`, del paquete `kernlab`. El vector `puntos[3:4]` es el conjunto de puntos en el espacio original ($\mathbb{R}^2$). Elegimos un kernel gaussiano con `kernel = "rbfdot"` y 3 dimensiones.

```
kernPCA <- kpca(x = as.matrix(scale(x = puntos[3:4], center = TRUE, scale = TRUE)),
                kernel = "rbfdot",
                kpar = list(sigma=0.1),
                features = 3
                )
```

Elegimos como parámetro $\sigma = 0{,}1$ mediante `kpar = list(sigma=0.1)`. La figura 5 muestra el mapeo de los puntos originales mapeados por kernel PCA, en $\mathbb{R}^3$ (`features=3`). Se puede ver que los puntos se representan en conjuntos separados por alturas, según el radio original.

```
# Kernel PCA
kernPCA <- kpca(x = as.matrix(scale(x = puntos[3:4], center = TRUE, scale = TRUE)),
                kernel = "rbfdot",
                kpar = list(sigma=.1),
                features = 3
                )

datosKPCA <- data.frame(rotated(kernPCA))
colnames(datosKPCA) = c("X", "Y", "Z")
radio <- cut(x = datosKPCA$Z, breaks = c(-6, -2, 2, 4),
             labels = c("Círculo 1", "Círculo 2", "Círculo 3"))
datosKPCA <- data.frame(datosKPCA, radio)

scatter3D(datosKPCA$X, datosKPCA$Y, datosKPCA$Z, bty = "b2", phi=25)
```

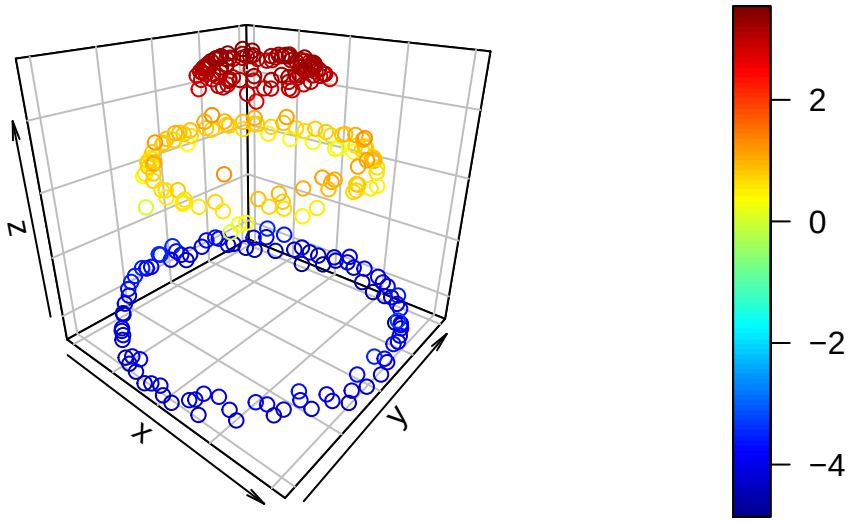

Figura 5: Resultado de KPCA aplicado a conjunto de círculos concéntricos, con kernel gaussiano de $\sigma = 0{,}1$.

Las varianzas acumuladas son

```
round(100*cumsum(eig(kernPCA))/sum(eig(kernPCA)), 2)
```

```
## Comp.1 Comp.2 Comp.3
##  44.11  87.74 100.00
```

Para una representación en 3 dimensiones se captura toda la variabilidad de las características, y con 2 dimensiones se captura aproximadamente un 88 % de la variabilidad.

La figura 6 muestra una representación de los puntos en el espacio de características, según sus dos componentes principales más relevantes. Coloreamos los puntos según el valor de su coordenada en el tercer eje factorial,

discretizando mediante la función `cut()` (los valores que se obtienen originalmente son continuos).

Con la configuración de kernel elegida, no se podrían separar linealmente, al igual que con un PCA común. Sin embargo, se pueden identificar los grupos de puntos (concretamente, sus proyecciones en el plano de los dos primeros ejes factoriales, que forman circunferencias concéntricas), incorporando información de la tercera coordenada como vemos en la figura siguiente:

```
ggplot(datosKPCA, aes(X, Y, colour=radio)) + geom_point() + coord_fixed()
```

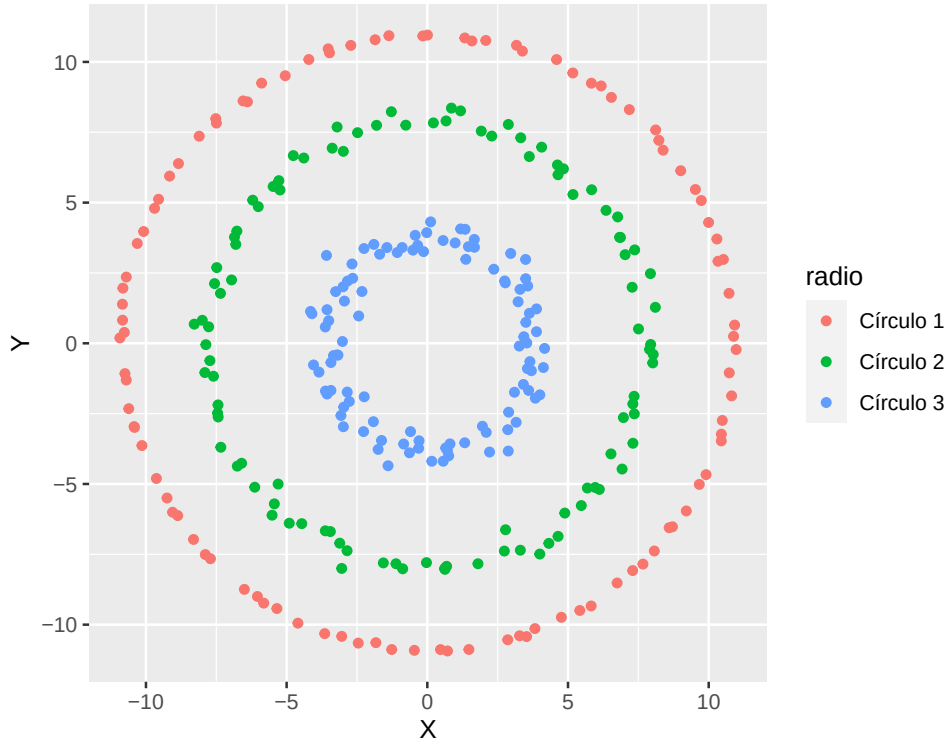

Figura 6: Clasificación de la representación de las primeras dos componentres, según el valor de la tercera componente obtenida.

En este caso, generamos una representación únicamente en 2 dimensiones:

```
# Kernel PCA
kernPCA <- kpca(x = as.matrix(scale(x = puntos[3:4], center = TRUE, scale = TRUE)),
                kernel = "rbfdot",
                kpar = list(sigma=1),
                features = 2
                )
```

La figura 7 muestra el resultado obtenido mediante kernel PCA la configuración anterior.

En este caso, es posible encontrar una separación lineal en tan solo dos dimensiones y sin recurrir a información de una tercera dimensión.

```
# Kernel PCA
kernPCA <- kpca(x = as.matrix(scale(x = puntos[3:4], center = TRUE, scale = TRUE)),
                kernel = "rbfdot",
                kpar = list(sigma=1),
                features = 2
                )
```

```
datosKPCA <- data.frame(rotated(kernPCA))
colnames(datosKPCA) = c("X", "Y")
cfa <- cut(x = datosKPCA$X, breaks = c(-10, -2.5, 5, 7.5),
           labels = c("Círculo 1", "Círculo 2", "Círculo 3"))
datosKPCA <- data.frame(datosKPCA, cfa)


ggplot(datosKPCA, aes(X, Y, colour=cfa)) + geom_point() + coord_fixed()
```

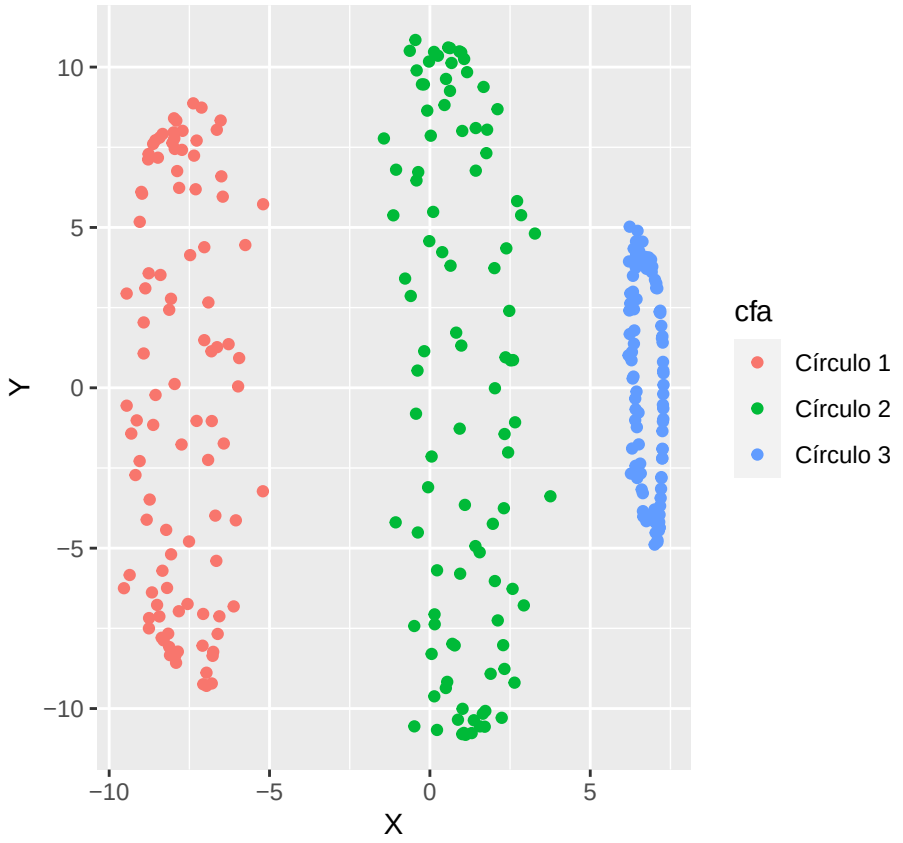

Figura 7: Resultado de KPCA aplicado a conjunto de círculos concéntricos, con kernel gaussiano de $\sigma = 1$.

Referencias

[1] Matias Lens. *Curso de Estadística Multivariada Computacional 2020*. Apuntes de clase, práctico. 2020.

[2] Rita Osadchy. «LECTURE: KERNEL PCA». En: *Unsupervised Learning 2011*. 2011. URL: https://bit.ly/37pQVJb.