Kernel PCA

PCA es una técnica de **aprendizaje** no supervisado, se utiliza para identificar tendencias en un conjunto de datos multidimensionales y reducir su dimensionalidad (variables), transformando esas variables (posiblemente correlacionadas, indicativo de existencia de información redundante) en un conjunto más pequeño de nuevas variables denominado «**Componentes principales**» y expresarlos de tal manera que resalten sus similitudes y diferencias, tratando de perder la menor cantidad de información posible (varianza). El nuevo conjunto de variables, componentes principales, son una función lineal de las variables originales, no correlacionadas y en un nuevo sistema de coordenadas, éstos pueden utilizarse a su vez en métodos de aprendizaje supervisado, como regresión de los componentes principales.

Matemáticamente existen varias formas de implementar el calculo, aquí partimos del caso de que las unidades de medida empleadas en las variables son las mismas, también de que todas las variables tienen una media de 0 y una desviación estándar 1, en caso contrario las variables con mayor varianza dominarían.

El proceso se realiza mediante el cálculo de la matriz de covarianza del conjunto de datos, despues se realiza el cálculo de los «Eigen» vectores y valores de la matriz, despues del cálculo se clasifican los valores propios de forma decreciente (ignorando los valores muy pequeños), de tal forma que la mayor varianza de cualquier proyección de los datos llega a ser la primera coordenada, la segunda más grande la segunda coordenada y así sucesivamente.

Kernel PCA

KPCA (Kernel principal component analysis) es una extensión de PCA que utiliza los métodos kernel. Tomamos un kernel y transformamos las operaciones lineales de PCA en un nuevo espacio con una asociación no lineal. Esta técnica necesita una alta capacidad computacional y mucho más tiempo que PCA. KPCA tiene otro inconveniente, en el supuesto de que tengamos un conjunto grande de datos produciremos una matriz grande (matriz de Gram) con una necesidad alta de memoria.

Para resolver los problemas de rendimiento, Fan de Zizhu, Jinghua Wang, Baogen Xu y Pengzhi Tang propusieron en 2.014 el KPCA eficiente (EKPCA). El algoritmo usa la evaluación de correlación de características para determinar los patrones básicos y luego los usa para reconstruir el modelo KPCA, realizando la extracción de características y clasificación de muestras de prueba. Dicen: como hay muchos menos patrones básicos que las muestras de entrenamiento, la extracción de funciones EKPCA es más eficiente computacionalmente logrando un rendimiento de clasificación similar.

Kernel PCA

En estadística, la transformación de datos es algo natural cuando se piensa que un fenómeno puede ser explicado considerando nuevas relaciones. Ya sea utilizando conocimiento a priori o de manera empírica, es común pensar en transformar observaciones a espacios de diferente dimensionalidad que en la que se encuentran originalmente. Por ejemplo, en regresión suelen añadirse términos de interacción como

$$y \sim x \rightarrow y \sim x + x^2$$

en este caso a través de los residuales podríamos interpretar que la transformación se pudiera aproximar mejor al modelo que se estudia.

Kernel PCA

En el Análisis de Componentes Principales mediante Kernels (Kernel PCA), se hace una transformación implícita de los datos y sobre estos datos transformados se realiza PCA. Puede ser que al transformar los datos se capture de mejor manera la variabilidad en el conjunto de datos que se está analizando.

El método basado en kernels puede verse como una generalización del Análisis de Componentes Principales, donde se practica una técnica que ayuda a la simplificación del problema resultante de la generalización. Esta técnica consiste en la utilización del kernel. Puede verse esta generalización como una extensión del PCA a dominios no-lineales, y de donde la utilización del kernel ayuda a eliminar los inconvenientes de trabajar en el contexto nolineal.

Kernel PCA

Un ejemplo sencillo de métodos de kernel

Para ilustrar cómo funcionan los métodos de kernel en general, se presenta aquí un ejemplo de clasificación sencillo . Supóngase que se tiene un conjunto de datos de entrenamiento $\{(x_i, y_i)\}$ con $y_i \in \{-1, 1\}$, y se quiere clasificar un punto nuevo a la clase a la cual tenga menor distancia con la media de la clase.

Esta construccion geometrica puede ser formulada exclusivamente en terminos del producto punto entre los datos como se vera en los slides siguientes :

Kernel PCA

Sean m_+ y m_- el número de datos en cada una de las clases, la media de cada clase es entonces

$$c_+ = \frac{1}{m_+} \sum_{i|y_i=+1} x_i,$$

$$c_- = \frac{1}{m_-} \sum_{i|y_i=-1} x_i.$$

Se asigna a un nuevo punto x la clase cuya media está más cercana como en la Figura

Kernel PCA

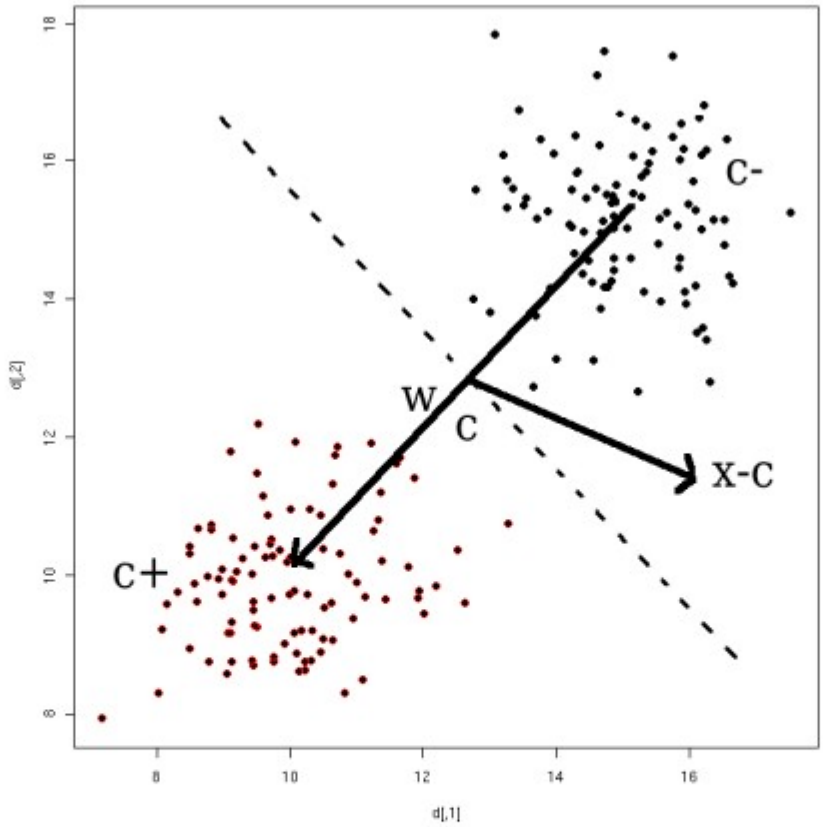

Fig 3.2

Figura : Ejemplo simple de clasificación. En la mitad del camino entre c_- y c_+ se encuentra el punto $c = (c_- + c_+)/2$. El vector $w = c_+ - c_-$ conecta las medias de las clases.

Kernel PCA

Esto lleva a considerar

$$\|c_- - x\|^2 \overset{?}{>} \|c_+ - x\|^2,$$

$$\|c_-\|^2 + \cancel{\|x\|^2} - 2\langle x, c_-\rangle \overset{?}{>} \|c_+\|^2 + \cancel{\|x\|^2} - 2\langle x, c_+\rangle,$$

$$\|c_-\|^2 - \|c_+\|^2 + 2\langle x, c_+\rangle - 2\langle x, c_-\rangle \overset{?}{>} 0,$$

$$y = signo(\langle x, c_+\rangle - \langle x, c_-\rangle + b), \tag{3.1}$$

Kernel PCA

donde se ha definido

$$b = \frac{1}{2}(\|c_-\|^2 - \|c_+\|^2)$$

con la norma $\|x\| = \sqrt{\langle x, x \rangle}$. Es instructivo reescribir la ecuación (3.1) en términos de los datos de entrenamiento x_i. Para ello, se susituyen en (3.1) las definiciones previamente mencionadas de c_- y c_+ con lo que se obtiene la función de decisión:

$$y = signo(\frac{1}{m_+} \sum_{i|y_i=+1} \langle x, x_i \rangle - \frac{1}{m_-} \sum_{i|y_i=-1} \langle x, x_i \rangle + b), \tag{3.2}$$

donde b es

$$b = \frac{1}{2}(\frac{1}{m_+^2} \sum_{(i,j)|y_i=y_j=+1} \langle x_i, x_j \rangle - \frac{1}{m_-^2} \sum_{(i,j)|y_i=y_j=-1} \langle x_i, x_j \rangle). \tag{3.3}$$

Kernel PCA

Cuando se tienen situaciones como la de la Figura (3-2) este algoritmo funciona razonablemente bien. Sin embargo, si se presenta una situación como la de la Figura (3-3) el algoritmo fracasa, puesto que en ese caso las medias prácticamente coinciden. Igual al ejemplo de regresión en el inicio de este capítulo, se puede llevar los datos de la Figura (3-3) a un espacio de dimensión mayor donde las medias de las clases no coincidan y el algoritmo pueda funcionar. Utilizar el *truco del kernel* en esta situación implica poder hacer tal transformación de forma indirecta. El requisito escencial para poder usar el truco del kernel es que un algoritmo pueda ser formulado de manera tal que los datos solamente aparezcan en forma de productos punto

Kernel PCA

Entonces, todas las ocurrencias de los productos punto se sustituyen por un *kernel*. De manera informal, un kernel es una función que regresa el valor del producto punto entre las imágenes de los dos argumentos:

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle .$$

Dada una funcion k, es posible verificar si es un kernel. Como ilustración considérese el *kernel polinomial*:

$$k(x, y) = (\gamma \langle x, y \rangle + c)^d, \gamma > 0. \tag{3.4}$$

Kernel PCA

Se puede demostrar que este kernel corresponde a un mapeo Φ a un espacio de características que es generado por todos los productos de d entidades de un patrón de entrada, por ejemplo, para $N = 2,\ d = 2,\ c = 0$ y $\gamma = 1$:

$$(x \cdot y)^2 = x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2,$$

$$(x \cdot y)^2 = (x_1^2, \sqrt{2}x_1 x_2, x_2^2) \cdot (y_1^2, \sqrt{2}y_1 y_2, y_2^2) = \Phi(x) \cdot \Phi(y)$$

el espacio implícito es $\Phi :\ (x_1^2, \sqrt{2}x_1 x_2, x_2^2)$. Así, sustituir las ocurrencias de productos punto por un kernel corresponde a realizar *una transformación implícita de los datos* al espacio Φ. Para el ejemplo de clasificación con medias, se puede sustituir en (3.2) y (3.3)

$$\langle x, y \rangle \rightarrow k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle ,$$

Formulacion del metodo Kernel PCA

Kernel PCA es el resultado de aplicar el truco del kernel al Analisis de Componentes Principales (PCA). Sabemos por lo presentado en la clase anterior que PCA es una transformacion de base para diagonalizar (encontrar eigenvalores-eigenvectores) del estimador de la matriz de covarianza

La matriz de covarianza se puede estimar como

$$C = \frac{1}{m}\sum_{j=1}^{m}(x_j - \hat{\mu})(x_j - \hat{\mu})^T$$

y la media como

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i.$$

Kernel PCA

Las nuevas coordenadas en la base de eigenvectores (las proyecciones ortogonales) son los Componentes Principales. Con Kernel PCA, se pretende generalizar este contexto a uno nolineal de la forma siguiente: supóngase que se hace el mapeo de los datos de forma nolineal a un *espacio de características* $\mathcal{H}$ mediante

$$\Phi : \mathbb{R}^N \to \mathcal{H},\ x \mapsto \Phi(x).$$

El método se basa en que para ciertas elecciones de Φ, aún si $\mathcal{H}$ tiene una dimensionalidad arbitrariamente grande, aún se puede realizar PCA en $\mathcal{H}$.

Supóngase que los datos mapeados al espacio de características $\Phi(x_1), \ldots, \Phi(x_m)$ están centrados, es decir, $\sum_{k=1}^{m} \Phi(x_k) = 0$ (suposición que más tarde será eliminada). Para hacer PCA cuando la estimación de la matriz de covarianza es

$$C = \frac{1}{m} \sum_{j=1}^{m} \Phi(x_j)\Phi(x_j)^T \tag{3.5}$$

se debe encontrar eigenvalores $\lambda \geq 0$ y eigenvectores $v \in \mathcal{H}$ que satisfagan

$$\lambda v = Cv. \tag{3.6}$$

Kernel PCA

Se puede observar que debido a (3.5) y (3.6) todas las soluciones v están en el espacio generado por $\Phi(x_1), \ldots, \Phi(x_m)$:

$$v = \frac{1}{\lambda m} \sum_{j=1}^{m} \Phi(x_j) \ \langle \Phi(x_j), v \rangle \, .$$

Esto es equivalente a decir que existen coeficientes $\alpha_1, \ldots, \alpha_m$ tales que

$$v = \sum_{i=1}^{m} \alpha_i \Phi(x_i). \tag{3.7}$$

Ahora bien, gracias a lo anterior se puede considerar el siguiente sistema equivalente a (3.6)

$$\lambda(\Phi(x_k) \cdot v) = (\Phi(x_k) \cdot Cv). \tag{3.8}$$

para todo $k = 1, \ldots, m$.

Kernel PCA

Sustituyendo en la expresión (3.8) las expresiones (3.5) y (3.7):

$$\lambda(\Phi(x_k) \cdot \sum_{i=1}^{m} \alpha_i \Phi(x_i)) = (\Phi(x_k) \cdot \frac{1}{m} \sum_{j=1}^{m} \Phi(x_j)\Phi(x_j)^T \sum_{i=1}^{m} \alpha_i \Phi(x_i))$$

$$\lambda \sum_{i=1}^{m} \alpha_i \langle \Phi(x_k), \Phi(x_i) \rangle = \frac{1}{m} \sum_{i=1}^{m} \alpha_i \left\langle \Phi(x_k), \sum_{j=1}^{m} \Phi(x_j) \langle \Phi(x_j), \Phi(x_i) \rangle \right\rangle$$

para todo $k = 1, \ldots, m$.

Definiendo la matriz K de Gram, de $m \times m$

$$K_{ij} = (\Phi(x_i) \cdot \Phi(x_j)) \tag{3.9}$$

Kernel PCA

y acomodando términos se llega a:

$$m\lambda K\alpha = K^2\alpha \tag{3.10}$$

donde α denota el vector columna con entradas $\alpha_1, \ldots, \alpha_m$. Para encontrar las soluciones de (3.10) se resuelve el problema de eigenvalores dado por

$$m\lambda\alpha = K\alpha$$

porque todas las soluciones de interés para la expresión anterior satisfacen (3.10) (ver [9]).

Con esto, se puede encontrar los eigenvectores $\{v^k\}$ en $\mathcal{H}$ a partir de la combinación lineal dada por (3.7). Para obtener eigenvectores normalizados en $\mathcal{H}$ se requiere que $v^k \cdot v^k = 1$. Esto se traduce que α debe cumplir:

$$1 = \sum_{i,j=1}^{m} \alpha_i^k \alpha_j^k (\Phi(x_i) \cdot \Phi(x_j)) = (\alpha^k \cdot K\alpha^k) = \lambda_k(\alpha^k \cdot \alpha^k). \tag{3.11}$$

Kernel PCA

Para obtener los componentes principales en $\mathcal{H}$, se calcula la proyección de un punto $(\Phi(x)$ en los eigenvectores v^k mediante

$$(v^k \cdot \Phi(x)) = \sum_{i=1}^{m} \alpha_i^k (\Phi(x) \cdot \Phi(x_i)). \tag{3.12}$$

Ahora bien, nótese que ni en (3.9) ni en (3.12) se requiere $\Phi(x)$ en forma explícita, sino solo en forma de productos punto. Entonces, la propuesta es utilizar funciones de kernel para calcular estos productos punto sin realizar el mapeo Φ.

Kernel PCA

Así, sustituyendo con kernels todas las ocurrencias de $\Phi(x) \cdot \Phi(y)$ se llega al siguiente algoritmo:

Algoritmo Kernel PCA:

- Calcular la matriz de Gram K (3.9)
- Diagonalizar K (encontrar $K = V\Lambda V^T$, donde V es la matriz que tiene como columnas los eigenvectores de K y Λ es una matriz diagonal con los correspondientes eigenvalores)
- Normalizar los coeficientes α de expansión de eigenvectores v (usando (3.11))
- Extraer l componentes principales, calculando las proyecciones de los datos sobre los primeros l eigenvectores (usando (3.12))

El supuesto bajo el que trabaja este algoritmo es el uso del estimador (3.5).

Kernel PCA

Hasta este punto, se ha supuesto que los $\Phi(x_i)$ están centrados en el espacio $\mathcal{H}$. Como no se puede en general centrar los datos (no se puede calcular la media de un conjunto de datos que no se tiene en forma explícita) se siguen los pasos anteriores usando $\tilde{\Phi}(x) = \Phi(x) - (1/m)\sum_{i=1}^{m}\Phi(x_i)$. Los elementos de la matriz de Gram son ahora

$$\widetilde{K_{ij}} = \tilde{\Phi}(x_i)\cdot\tilde{\Phi}(x_j) = (\Phi(x_i) - \frac{1}{m}\sum_{r=1}^{m}\Phi(x_r))\cdot(\Phi(x_j) - \frac{1}{m}\sum_{s=1}^{m}\Phi(x_s))$$

$$\widetilde{K_{ij}} = (\Phi(x_i)\cdot\Phi(x_j) - \frac{1}{m}\sum_{r=1}^{m}\Phi(x_j)\cdot\Phi(x_r) - \frac{1}{m}\sum_{s=1}^{m}\Phi(x_i)\cdot\Phi(x_s) +$$

$$+\frac{1}{m^2}\sum_{r=1}^{m}\sum_{s=1}^{m}\Phi(x_r)\cdot\Phi(x_s)).$$

Kernel PCA

Definiendo la matriz 1_m de $m \times m$ cuyas entradas son todas $1/m$, se llega a que la matriz que se debe diagonalizar (en lugar de K) para el algoritmo de Kernel PCA se puede expresar en términos de K misma como

$$\widetilde{K_{ij}} = K - 1_m K - K 1_m + 1_m K 1_m.$$

Kernel PCA

[1] N. A. Campbell. Robust procedures in multivariate analysis I: robust covariance estimation. Appl. Statist., 29(3):231–237, 1980.

[2] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. Robust Statistics. The Approach Based On Influence Functions. Wiley Series In Probability And Mathematical Statistics. John Wiley & Sons, 1985.

[3] T. Hastie, R. Tibshirani, and J. Friedman. The Elements Of Statistical Learning. Springer-Verlag, 2001

[4] P. J. Huber. Robust Statistics. Wiley Series In Probability And Mathematical Statistics. John Wiley & Sons, 1981.

[5] M. Hubert, P.J. Rousseeuw, and K. Vanden Branden. Robpca: a new approach to robust principal component analysis. Technometrics, 2005.

[6] J. E. Jackson. A User's Guide To Principal Components. Wiley Series In Probability And Mathematical Statistics. John Wiley & Sons, 1991.

[7] I. T. Jolliffe. Principal Component Analysis. Springer Series In Statistics. Springer-Verlag, 1986.

Kernel PCA

[8] P. J. Rousseeuw and A. M. Leroy. Robust Regression and Outlier Detection. Wiley Series In Probability And Mathematical Statistics. John Wiley & Sons, 1987.

[9] B. Scholkopf and A. Smola. Learning With Kernels. MIT Press, 2002.

[10] B. Scholkopf, A. Smola, and K. R. M¨uller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, pages 1299–1319, 1998.

[11] J. Shawe-Taylor and N. Cristianini. Kernel Methods For Pattern Analysis. Cambridge University Press, 2004.

[12] F. Torre and M. Black. Robust principal component analysis for computer vision, 2001.