IIC2433 — Minería de Datos

Apuntes: Reducción de dimensionalidad

1. Por qué reducir la dimensionalidad

Recordemos que la tarea de reducir la dimensión puede describirse, a grandes rasgos, como la tarea de llevar un dataset X de vectores en d dimensiones a un dataset Y en donde cada vector ahora es de $k < d$ dimensiones, y tal que X e Y sean *suficientemente similares*. Por supuesto, la noción de qué es ser suficientemente similares nos va a llevar a distintas nociones de reducción de dimensionalidad. Por ejemplo:

- Si pensamos que Y es una proyección de X en algún hiperplano, entonces querríamos que los vectores de X e Y estén lo más cerca posible (de acuerdo a alguna métrica de distancia).
- Podríamos querer que Y fueran 2 o 3 dimensiones, de forma de poder visualizar X de la mejor forma.
- Si vamos a clusterizar, nos gustaría que Y mantuviera los vecinarios de X.

Pero también podemos ver esto de un modo más abstracto: como ya decíamos, quizás Y captura un espacio *latente* que es el que más información tiene para una cierta tarea de clasificación, y al pasar a Y nos concentramos solo en este espacio de alta información, lo que nos puede ayudar a reducir el overfitting.

Y, por supuesto, hay razones técnicas o computacionales. Por ejemplo:

- Si nuestro algoritmo realiza computos matriciales o computos sobre todas las dimensiones, puede que sea imposible de realizar sobre todo el dataset, y hay que reducir la dimensionalidad.
- Si nuestro algoritmo se basa en distancias euclideanas, no va a funcionar bien en altas dimensiones por culpa de la *maldición de la dimensionalidad*.

1.1. La maldición de la dimensionalidad

Normalmente tendemos a pensar en 2d o 3d, donde las nociones de distancias tienen de cierta forma una componente espacial que entendemos bien. Pero cuando tratamos de aplicar la misma intuición para espacios de muchas dimensiones, esto ya no funciona!

Una forma simple de ver esto es lq siguiente. En un espacio de d dimensiones, el *hipervolumen* total de un hipercubo con aristas de tamaño unitario es 1, independiente de la

dimension. Pero en el caso de la esfera, la fórmula para su (hiper) volumen está dada por $\frac{\pi^{\frac{d}{2}}}{d\Gamma(\frac{d}{2}+1)}$, donde lo importante para nosotros que que la función Γ de Euler crece más rápido que $\pi^{\frac{d}{2}}$. Así, a medida que crecen las dimensiones, el hiper-volumen de la esfera de radio 1 tiende a cero!

A no ser que realmente sepamos lo que estamos haciendo, se recomienda no usar algoritmos que usen métricas de distancias como la Euclideana en datasets de alta dimensión, por que su comportamiento está lejos de ser intuitivo.

2. Principal Component Analysis (PCA)

Recuerda que trabajamos sobre d dimensiones. La premisa del PCA es simple: encontrar una matriz que proyecte a X a un hiper-plano de k dimensiones, de forma que la distancia entre los vectores sea la más pequeña. Hagamos esto más específico.

Importante: Para estos calculos usamos la notación que una matrix X con m entidades sobre d dimensones tiene d filas y m columnas (cada entidad es un vector columna).

2.1. Matrices de proyección y norma de matrices

El algoritmo de PCA se basa en encontrar la mejor matriz de proyección P, ortogonal y de rango k [1], de forma que PX esté lo más cerca posible de X. Intuitivamente, la matriz P (de rango k) está *proyectando* (o más bien transformando) linealmente los puntos en X a un hiper-plano de k dimensiones.

Para capturar la noción de *estar lo más cerca posible*, en PCA se usa la idea de que la matriz diferencia $PX - X$ tenga cada entrada lo más pequeña posible, lo que se captura minimizando

$$||A||_F = \sqrt{\sum_{i \leq d} \sum_{j \leq d} |a_{ij}|},$$

número conocido como la norme Euclideana o norma Frobenius.

Entonces, para reducir X de d dimensiones a $k < d$ dimensiones, la idea es encontrar la matriz ortogonal de proyección P^*, de rango k, que minimiza:

$$||PX - X||_F^2$$

2.2. Solución mediante matriz de covarianzas

Como encontramos esta matriz? Acá hay un algoritmo para hacerlo.

- Transformar X en X' restándole a cada vector en x la media o centro de gravedad: Si X tiene entidades $x^1, \ldots, x^m$ y $\mu = \frac{\sum x^j}{m}$, entonces X' es la matriz dada por las entidades $x^1 - \mu$, $x^2 - \mu$, etc.

[1] Las matrices de proyección sobre d dimensiones son matrices de dxd que satisfacen $P^2 = P$. Cuando la proyección es ortogonal, se tiene además que $P^T = P$

- Calcular $C = \frac{XX^T}{m}$, lo que se conoce como matriz de covarianza. Específicamente,

$$C_{ij} = \sum 1 \leq \ell \leq d X_{i\ell} X_{\ell j}$$

- Tomar los k vectores propios más relevantes (es decir, los que tienen asociados los k valores propios más altos). Les llamamos $v_1, \ldots, v_k$. La matriz $U_k = [v_1 \ldots v_k]$, con cada uno de esos vectores como columnas, es nuestra matriz para proyectar a las k componentes principales.

- La representación buscada es entonces $Y = U_k^T X$. Nota que es una matriz de k x m: ahora cada una de las entidades tiene asociado un vector de solo k dimensiones.

- La matriz de proyección P^* que minimiza el problema descrito en la sección anterior es $U_k U_k^T$.

Señalamos que usualmente este NO es el algoritmo empleado por las librerías que vamos a usar en clases. Lo usual es encontrar estos vectores mediante una técnica que se conoce como Singular Value Decomposition (SVD), pero eso queda fuera de lo que esperamos en este curso.