



# Selección y Reducción de atributos

Felipe Bravo

Basado ligeramente en slides previas de  
Benjamín Bustos

# Selección y Reducción de Atributos

Muchas veces tenemos atributos irrelevantes o redundantes en nuestros datos.

Vamos a ver dos enfoques para atacar este problema.

1. **Selección de atributos:** es un enfoque supervisado donde escogemos el subconjunto de atributos más útil para nuestro problema de clasificación.
2. **Reducción de atributos:** enfoque no-supervisado donde encontramos una proyección de menor dimensionalidad que concentra la información contenida en los datos.

# Selección de Atributos

- Es un proceso supervisado que busca encontrar el mejor subconjunto de atributos para una tarea de clasificación/regresión.
- La presencia de atributos irrelevantes o redundantes puede afectar el desempeño de un modelo.

# Selección de Atributos

Por ejemplo, agregar un atributo aleatorio (por ende irrelevante) afecta a los árboles de decisión.

- Esto es sorprendente puesto que los árboles están pensados para solo seleccionar atributos relevantes.
- Pero en la práctica es común que el árbol escoja al atributo irrelevante en algún momento del branching y “aprenda” el ruido.
- Piensen que cuando bajamos la profundidad del árbol el splitting se hace sobre menos datos.
- Entonces el atributo irrelevante podría separar las clases por chance en estos datasets pequeños.
- A esto se le llama **fragmentación**.

# Selección de Atributos

- KNN es muy sensible a atributos irrelevantes
  - Todos los atributos se ponderan igual en el cálculo de distancias.
  - La clasificación solo se hace sobre pocos datos (los vecinos más cercanos). ¡Un atributo irrelevante afecta mis distancias!
  - En KNN el número de datos de entrenamiento requeridos para realizar buenas clasificaciones aumenta exponencialmente con el número de atributos irrelevantes.

# Selección de Atributos

- Naïve Bayes si es robusto a atributos irrelevantes.
  - El valor de  $P(A|C)$  es parecido para todas las clases cuando A es irrelevante => el atributo irrelevante es implícitamente ignorado.
- Pero, naïve Bayes es sensible a atributos redundantes (pares de atributos fuertemente correlacionados entre sí).
  - Al asumir independencia condicional entre atributos se amplifica el efecto de los atributos redundantes sobre la clase.
  - Si A1 y A2 son redundantes, nuestras probabilidades posteriores tendrán el efecto amplificado  $P(A1|C)*P(A2|C)$
  - Por otro lado, un árbol tendería a escoger solo uno y descartar el otro.

# Enfoques

- Debido al efecto negativo de los atributos irrelevantes para varios métodos de aprendizaje, es común realizar una selección de atributos.
- Una selección manual es muy costosa.
- La selección automática se divide en dos enfoques:
  - a. **Scheme-independent o método de filtro:** evalúa el subconjunto de atributos en base a características generales de los datos.
  - b. **Scheme-dependent o método de wrapper:** evalúa el subconjunto de atributos usando un clasificador (la calidad de los atributos se define por la capacidad predictiva del modelo).

# Selección de atributos Scheme-independent

- Los atributos se seleccionan usando alguna métrica calculada a partir de los datos.
  - Ejemplo: Entropía, Mutual Information, Information Gain.
- También se pueden usar algoritmos de aprendizaje **rápidos** que entregan información sobre la utilidad de los atributos:
  - Los atributos escogidos en los nodos de más arriba en un árbol de decisión.
  - Los coeficientes de un modelo lineal con alto valor absoluto.
  - Esto es distinto al enfoque **wrapper** pues no consideramos la capacidad predictiva del modelo.



# Scheme-independent attribute selection

## Correlation-based Feature Selection (CFS):

- Encontrar atributos que correlacionan con la clase, pero que a la vez tienen poca correlación entre sí:
  - Se mide la correlación entre atributos categóricos usando symmetric uncertainty:

$$U(A, B) = 2 \frac{H(A) + H(B) - H(A, B)}{H(A) + H(B)} \in [0, 1]$$

- $H(A)$ ,  $H(B)$  son la entropía del atributo correspondiente.
- $H(A, B)$  es la entropía conjunta de  $A, B$ .

$$H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} P(x, y) \log_2 [P(x, y)]$$

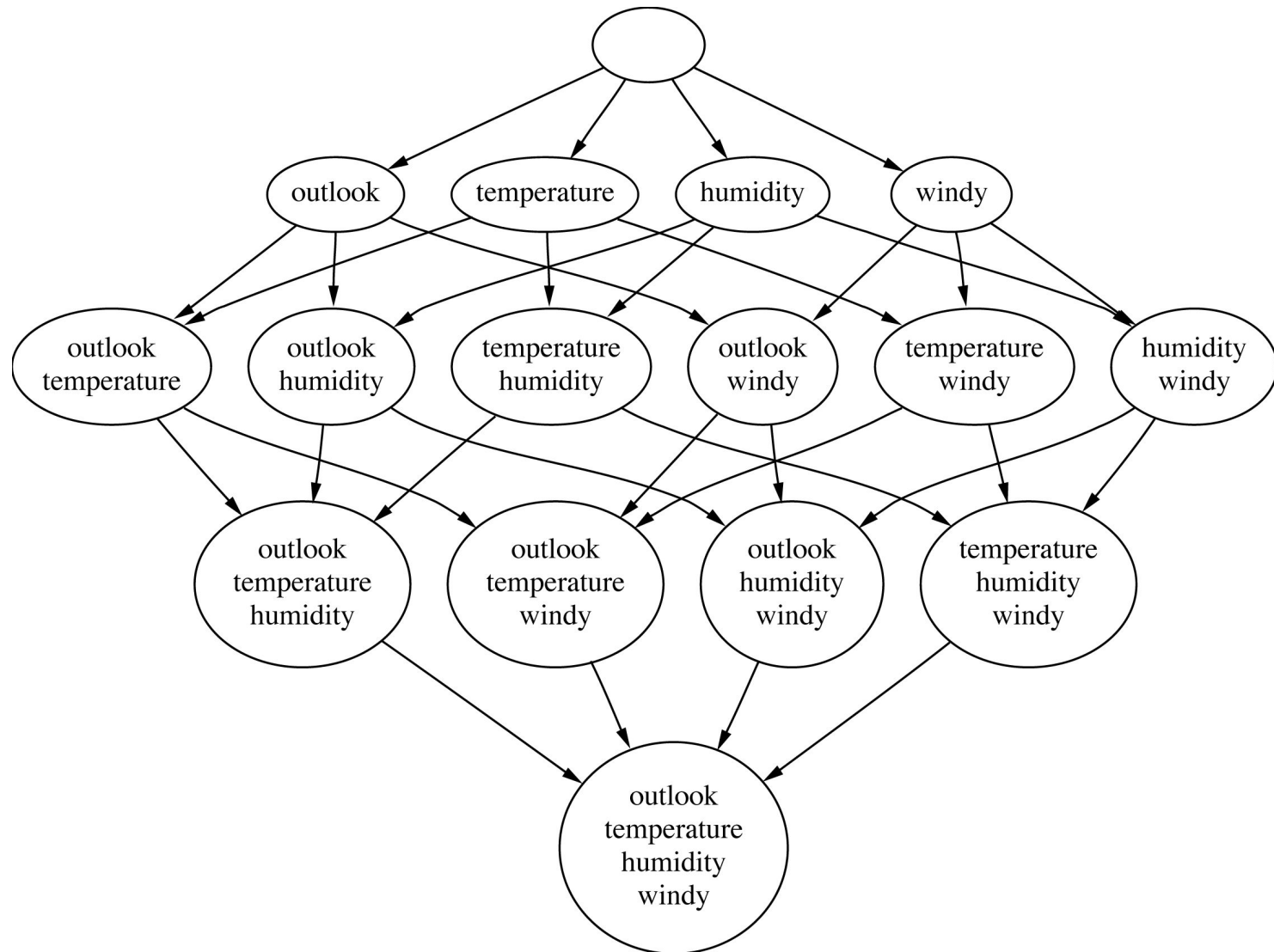
# Scheme-independent attribute selection

- CFS mide la calidad de un conjunto de atributos como:

$$\sum_j U(A_j, C) / \sqrt{\sum_i \sum_j U(A_i, A_j)}$$

- Favorece atributos con alta asociación con la clase pero penaliza pares de atributos con alta asociación entre sí.
- Favorece conjuntos más pequeños en caso de empates.

# Subconjuntos de atributos para el weather dataset



# Buscando en el espacio de atributos

- Necesito recorrer mi espacio de subconjuntos de atributos y evaluar cada candidato según mi “criterio”. El criterio puede venir de un método de filtro o de un método de wrapper.
- El número de subconjuntos posibles de atributos es exponencial sobre el número total de atributos.
- Algunas estrategias greedy (no aseguran encontrar el óptimo):
  - forward selection (top-down): parto con cero atributos y voy agregando atributos uno por uno escogiendo el que maximiza el criterio. Paro cuando no veo mejora al agregar atributos nuevos.
  - backward elimination (bottom-up): parto con todos los atributos y voy sacando atributos a medida que mejore mi criterio.

# Buscando en el espacio de atributos

- Estrategias más sofisticadas:
  - **Bidirectional search**: combina forward con backward elimination.
  - **Best-first search**: en vez de parar al no encontrar mejora (usando forward o backward search), mantiene una lista de los subconjuntos de atributos evaluados ordenados según el criterio.
    - Hay que asignarle algún condición de parada para que no recorra todo el espacio de búsqueda.
  - **Beam search**: aproximación truncada de best-first search. El ancho “width” del beam o “viga” define el número de candidatos a tener guardados en la lista.
  - **Genetic algorithms**: inspirados en el principio de selección natural. La selección de atributos “evoluciona” al combinar subconjuntos de atributos “buenos” según el criterio.

# Selección Scheme-specific ó Wrapper (Envoltura)

- Enfoque Wrapper: la selección se hace usando un clasificador sobre el subconjunto de atributos y evaluando su performance (accuracy, F1, AUC).
- El proceso de selección es caro computacionalmente.
- Si cada evaluación se hace con 10-fold cross-validation, debemos ejecutar el algoritmo de aprendizaje 10 veces.

# Selección Scheme-specific ó Wrapper (Envoltura)

- Con  $m$  atributos, una estrategia “greedy” (forward o backward) multiplica el tiempo de evaluación por un factor proporcional a  $m^2$  en el peor caso.
- Esto es mucho más caro para estrategias de búsqueda más sofisticadas, donde puede llegar a ser orden  $2^m$  para una búsqueda exhaustiva que evalúa los  $2^m$  subconjuntos posibles.
- ¡El enfoque wrapper se porta bien con Naive Bayes!

# Reducción de la dimensión

- Proceso de reducir el número de características en un dataset de atributos numéricos de manera no-supervisada.
- Ventajas
  - Elimina ruido y/o características redundantes.
  - Mejora la eficiencia del análisis de los datos.
  - Puede mejorar el rendimiento de un clasificador.
  - Útil para proyectar datos a dos dimensiones y poder visualizar tanto los datos como el resultado de método de clustering.
- Métodos a estudiar:
  - Análisis de Componentes Principales (PCA)
  - Multidimensional Scaling (MDS)



# Conceptos matemáticos

- Media de una población

$$\bar{x} = \frac{1}{m} \sum_{i=1}^m x_i$$

- Varianza

$$s^2 = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})^2$$

# Conceptos matemáticos

- Covarianza

$$cov(x, y) = \frac{1}{m-1} \sum_{i=1}^m (x_i - \bar{x})(y_i - \bar{y})$$

- Covarianza es conmutativa

- Signo de la covarianza:

- +: ambas dimensiones se incrementan juntas
- - : si una se incrementa, la otra se decrementa

# Conceptos matemáticos

- Matriz de covarianza
  - Para vectores  $n$ -dimensionales

$$C^{n \times n} = (c_{ij} | c_{ij} = \text{cov}(\text{Dim}_i, \text{Dim}_j))$$

- Ejemplo con vectores 3-dimensionales

$$C = \begin{pmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{pmatrix}$$

Para una matriz  $X$  de  $m$  vectores donde todos los vectores tienen media cero, se tiene que  $X^T X = m \Sigma$  donde  $\Sigma$  es la matriz de covarianza.

# Conceptos matemáticos

- Vectores y valores propios

- Se calculan para matrices cuadradas
- Sea **A** una matriz de  $n \times n$ , *esta* tiene  $n$  vectores propios **v** (si es que existen) que satisfacen

$$Av = \lambda v$$

- Donde  $\lambda$  es un escalar llamado valor propio.
- Propiedad: vectores propios son ortogonales entre sí.

# Conceptos matemáticos

- Vector propio normalizado (unitario): su largo es 1
- Cálculo de valores propios
  - Resolver sistema de ecuaciones lineales

$$\det(A - \lambda I) = 0$$

- Cálculo de los vectores propios

$$(A - \lambda I) \cdot v = 0$$

# PCA

- *PCA: Principal Component Analysis*
  - Sirve para encontrar patrones en los datos
  - Sirve para reducir su dimensión sin perder “much” información
    - Retiene características de los datos que contribuyen más a su varianza
- También conocida como la transformación Karhunen-Loève (KLT).

# Motivación

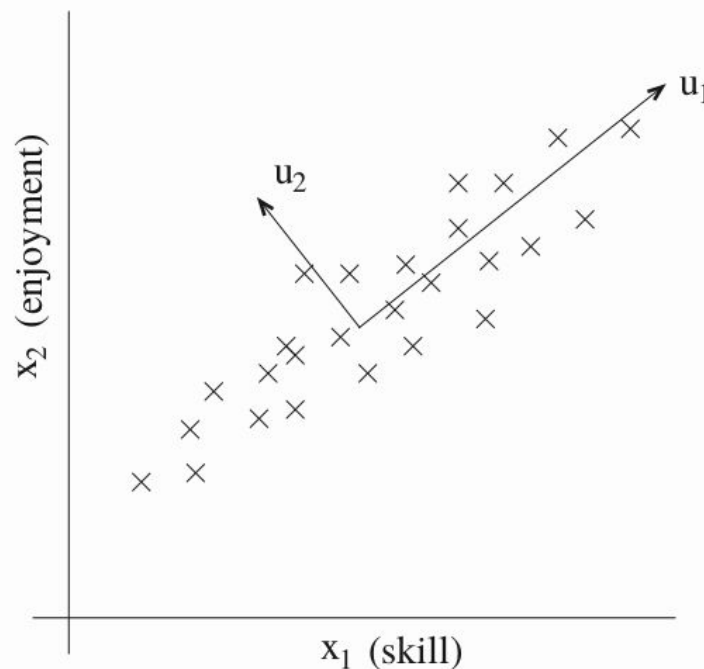
- Supongamos que tenemos un dataset de  $n$  atributos y  $m$  ejemplos de automóviles:

$$\{x^{(i)}; i = 1, \dots, m\}$$

- Atributos: máxima velocidad, radio de giro, etc..
- Supongamos que hay dos atributos que son la velocidad máxima medida en Km/h y otra en millas por hora, que son casi linealmente dependientes salvo por pequeños redondeos numéricos.
- ¿Cómo podemos detectar y ojalá eliminar esta redundancia?

# Motivación

- Otro ejemplos: Tenemos una encuesta hecha a pilotos de helicóptero.
- $X_1$  corresponde a qué tan habilidoso es el piloto y  $X_2$  corresponde a cuánto disfruta la actividad.
- Cómo ser un buen piloto requiere mucha dedicación es común que los buenos pilotos disfruten mucho de la actividad.





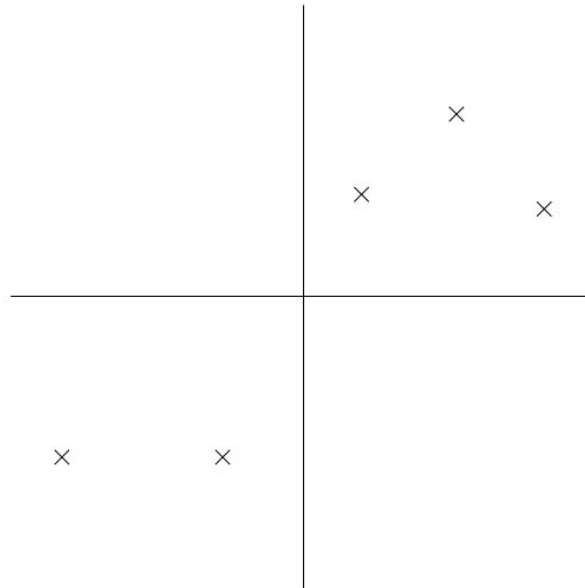
# Motivación

- $X_1$  y  $X_2$  están fuertemente correlacionados.
- De hecho uno podría plantear que los datos están sobre un eje diagonal (la dirección del vector  $u_1$ ) que capturan el “karma” intrínseco del piloto
- Luego  $u_2$  proyecta el ruido.
- ¿Cómo podemos calcular la dirección de  $u_1$  automáticamente?
- Antes de explicar PCA tenemos que normalizar los datos para que tengan media nula y varianza unitaria:

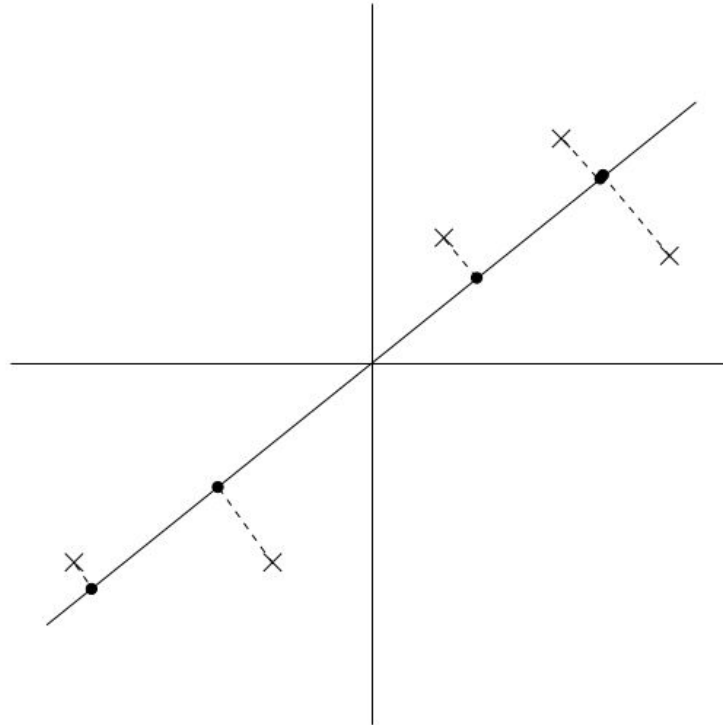
1. Let  $\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$ .
2. Replace each  $x^{(i)}$  with  $x^{(i)} - \mu$ .
3. Let  $\sigma_j^2 = \frac{1}{m} \sum_i (x_j^{(i)})^2$
4. Replace each  $x_j^{(i)}$  with  $x_j^{(i)} / \sigma_j$ .

# PCA

- Nuestro objetivo es encontrar un vector unitario  $u$  ( $\|u\| = 1$ ), tal que cuando proyectemos los datos al eje definido por  $u$  la varianza se maximice.
- Ejemplo: Sea el siguiente dataset

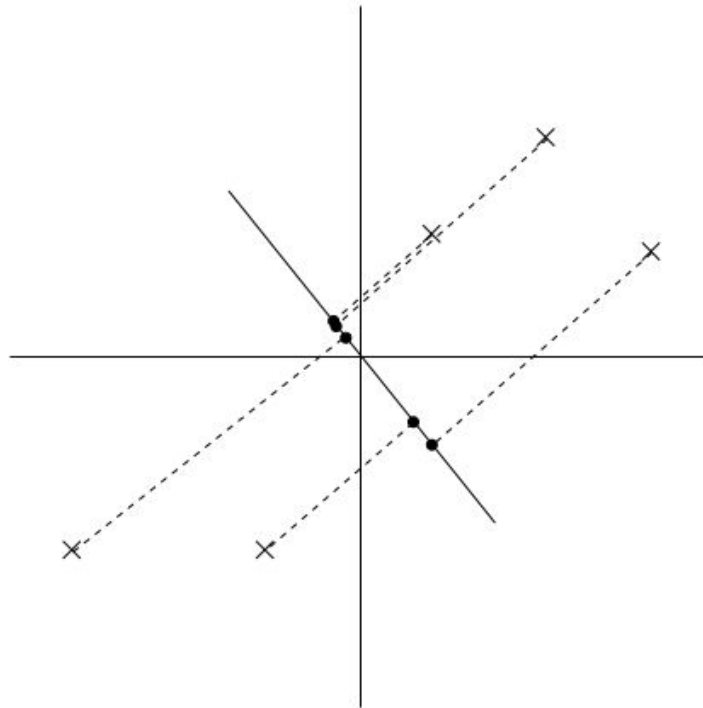


# PCA



- Los círculos reflejan la proyección de los datos originales sobre la línea.
- Los datos proyectados tienen alta varianza y están lejos de cero.

# PCA



- Si proyectos en otra dirección, los datos proyectados tienen mucho menos varianza y están más cerca del origen.

# PCA

- Nuestro objetivo es encontrar automáticamente la dirección  $\mathbf{u}$  que proyecta los datos a una máxima varianza.
- Sea  $\mathbf{u}$  un vector unitario y otro vector  $\mathbf{v}$ .
- Por el álgebra lineal sabemos que  $\mathbf{u} \cdot \mathbf{v} = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\theta)$
- Esto se puede reordenar como:

$$\mathbf{u} \cdot \mathbf{v} = (\|\mathbf{v}\| \cos(\theta)) \|\mathbf{u}\| = \mathbf{v}_u \|\mathbf{u}\|$$

- Donde  $\mathbf{v}_u = \|\mathbf{v}\| \cos(\theta)$  , representa el largo de  $\mathbf{v}$  en la dirección de  $\mathbf{u}$ .

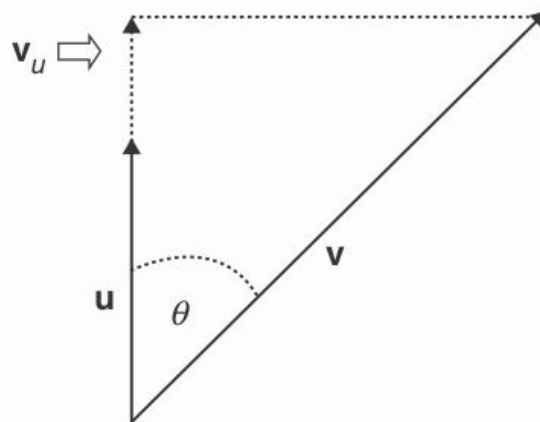


Figure A.2. Orthogonal projection of vector  $\mathbf{v}$  in the direction of vector  $\mathbf{u}$ .

# PCA

- Como  $\mathbf{u}$  es vector unitario  $\Rightarrow \|\mathbf{u}\| = 1$
- Entonces si  $\mathbf{u}$  es un vector unitario, el producto punto  $\mathbf{u} \cdot \mathbf{v}$  es la proyección del vector  $\mathbf{v}$  en  $\mathbf{u}$ .
- A esto se le llama la proyección ortogonal de  $\mathbf{v}$  en  $\mathbf{u}$ .

# PCA

- Entonces, sea  $\mathbf{u}$  un vector unitario y  $\mathbf{x}$  un ejemplo de nuestro dataset
- Por lo visto anteriormente sabemos que la proyección de  $\mathbf{x}$  sobre  $\mathbf{u}$  se puede calcular como  $\mathbf{x}^T \mathbf{u}$ .
- Entonces para maximizar la **varianza de la proyección** tenemos que encontrar un vector unitario  $\mathbf{u}$  que maximice la siguiente ecuación:

$$\begin{aligned}\frac{1}{m} \sum_{i=1}^m (x_{(i)}^T \cdot u)^2 &= \frac{1}{m} (Xu)^T (Xu) \\ &= \frac{1}{m} u^T X^T X u = u^T \frac{X^T X}{m} u \\ &= u^T \Sigma u\end{aligned}$$

Donde  $\Sigma$

es la matriz de covarianza asumiendo que los datos tiene media nula.

# PCA

- Maximizar la ecuación anterior sujeto a que  $\|u\|_2=1$  nos da el vector propio principal de la matriz de covarianza  $\Sigma$ .
- La restricción  $\|u\|_2=1$  equivale a decir que  $u^T u = 1$
- Entonces queremos resolver el siguiente problema:

$$\begin{aligned} & \max_u u^T \Sigma u \\ & \text{sujeto a } u^T u = 1 \end{aligned}$$

Esto usando multiplicadores de Lagrange equivale a:

$$\mathcal{L}(u, \lambda) = u^T \Sigma u - \lambda(u^T u - 1)$$

OJO:  $\Sigma$  es la matriz de covarianza (no es una sumatoria)



# PCA

Maximizamos  $\mathbf{u}$ , derivando e igualando a cero:

$$\nabla_{\mathbf{u}} \mathcal{L} = \Sigma \mathbf{u} - \lambda \mathbf{u}$$

Y eso nos da:  $\Sigma \mathbf{u} = \lambda \mathbf{u}$

- Esta es exactamente la ecuación para encontrar el vector propio de  $\Sigma$ .
- El vector propio principal (de máxima varianza) se asocia al mayor valor propio  $\lambda$ .
- Si queremos reducir mis datos a  $k$  dimensiones, escojo los vectores propios  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k$  de mi matriz de covarianza asociados a los  $k$  valores propios más grandes.
- Los vectores propios son ortogonales entre sí.

# PCA

- Algoritmo (entrada: vectores en  $\mathbb{R}^d$ ):
  1. Centrar los datos y normalizar los datos: restar la media en cada dimensión y normalizar por la desviación estándar.
  2. Calcular la matriz de covarianza

$$\hat{C} = \begin{pmatrix} cov(\hat{x}, \hat{x}) & cov(\hat{x}, \hat{y}) & cov(\hat{x}, \hat{z}) \\ cov(\hat{y}, \hat{x}) & cov(\hat{y}, \hat{y}) & cov(\hat{y}, \hat{z}) \\ cov(\hat{z}, \hat{x}) & cov(\hat{z}, \hat{y}) & cov(\hat{z}, \hat{z}) \end{pmatrix}$$

# PCA

- Algoritmo:
  3. Calcular valores y vectores propios (normalizados) de la matriz de covarianza
  4. Elegir componentes principales
    - Ordenar valores propios en orden descendente
      - Primer componente principal: vector propio asociado al valor propio mayor
      - Segundo componente principal: vector propio asociado al segundo valor propio mayor
      - Etc.

# PCA

- Algoritmo:

4. Transformada lineal:

$$W = (eig_1; eig_2; \dots; eig_d)$$

5. Transformación de los datos:

$$y = W^T \cdot \hat{x}$$

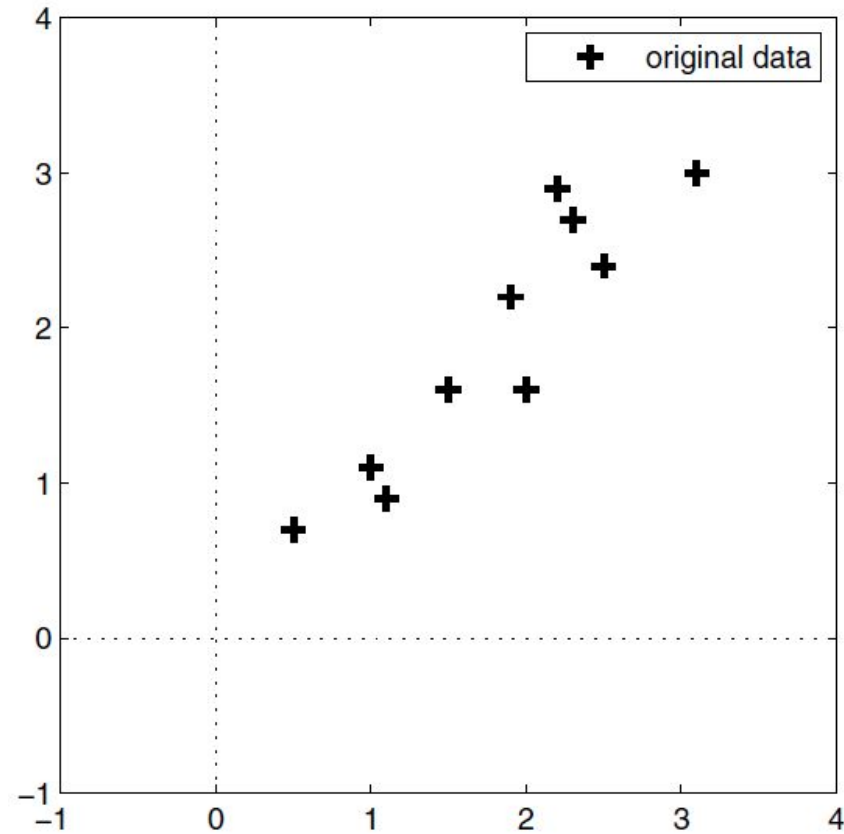
# PCA

- Reconstrucción de los datos

$$x = \left( (W^T)^{-1} \cdot y \right) + \bar{x} == \left( (W^T)^T \cdot y \right) + \bar{x} = (W \cdot y) + \bar{x}$$

En matrices ortogonales (como W) la inversa es equivalente a la transpuesta.

# Ejemplo



**Fig. 3.10.** Original PCA data plot.

# Ejemplo

- Cálculo de matriz de covarianza

$$\hat{\Sigma} = \begin{pmatrix} 0.61656 & 0.61544 \\ 0.61544 & 0.71656 \end{pmatrix}$$

- Notar correlación positiva entre ambas dimensiones

# Ejemplo

- Valores propios:

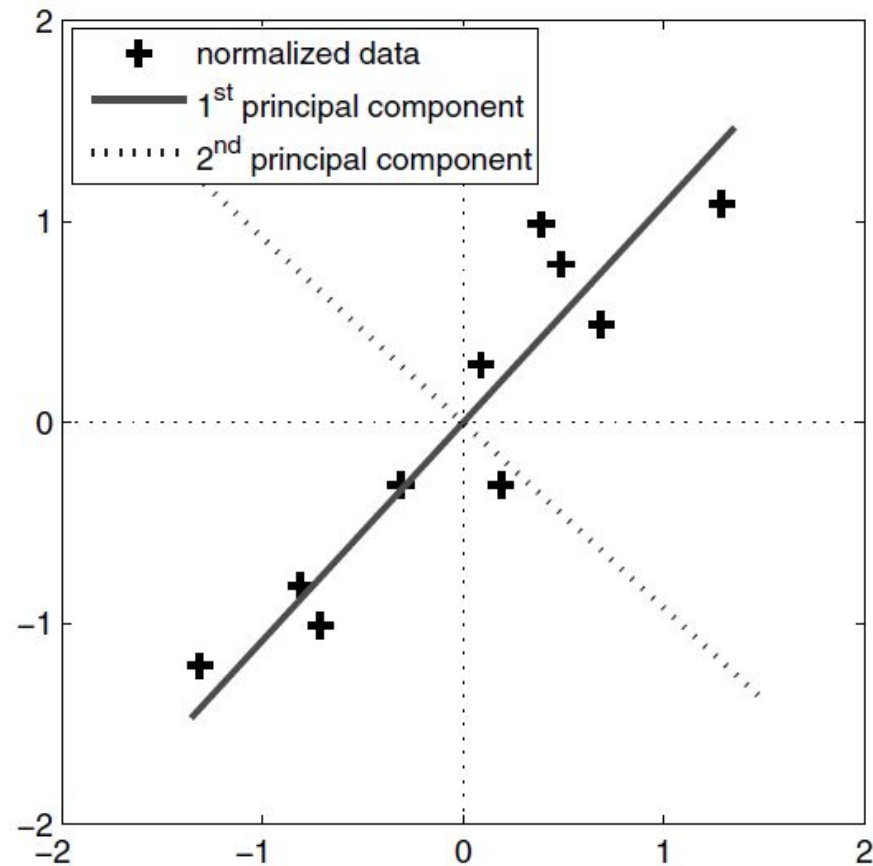
$$\lambda_1 = 0.049083 \text{ and } \lambda_2 = 1.284$$

- Vectores propios (columnas):

$$\begin{pmatrix} -0.73518 & -0.67787 \\ 0.67787 & -0.73518 \end{pmatrix}$$



# Ejemplo



**Fig. 3.11.** Normalized PCA data and the principal components.

# Ejemplo

- Eligiendo componentes principales:

$$W = (w_1; w_2) = \begin{pmatrix} -0.67787 & -0.73518 \\ -0.73518 & 0.67787 \end{pmatrix}$$

- Transformar datos

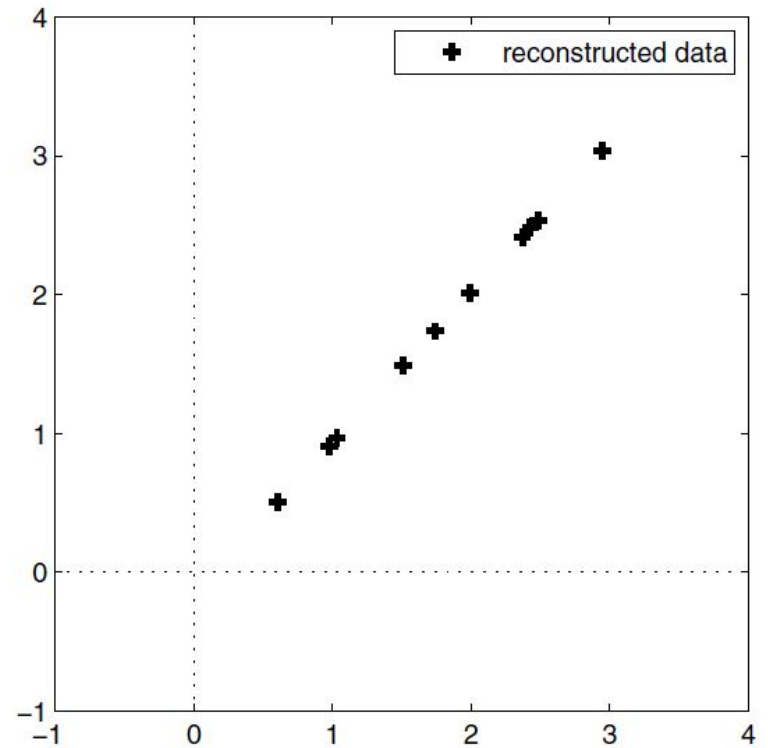
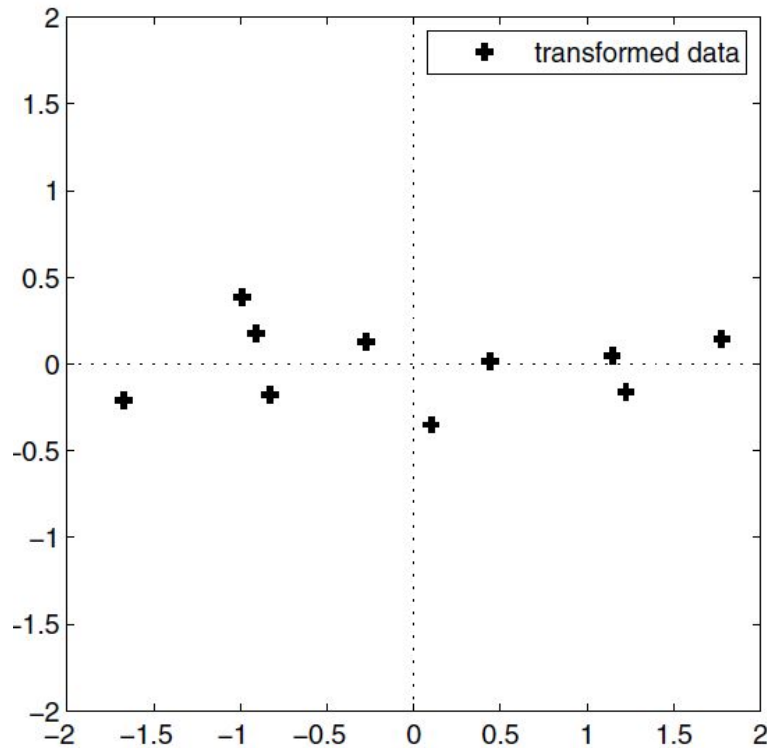
$$y = W^T \cdot \hat{x}$$

# Reducción de dimensión

- Se eligen sólo las  $k$  componentes principales más significativas para formar la matriz de transformación
  - Esto reduce la dimensión de los datos a  $k$
- Para el ejemplo: reducir a una dimensión

$$W = (w_1) = \begin{pmatrix} -0.67787 \\ -0.73518 \end{pmatrix}$$

# Reducción de dimensión



**Fig. 3.12.** *Left:* transformed PCA data plot; *right:* reconstructed data transformed using a single principal component.

# Implementación en Numpy

```
1 from numpy import array
2 from numpy import mean
3 from numpy import cov
4 from numpy.linalg import eig
5 # define a matrix
6 A = array([[1, 2], [3, 4], [5, 6]])
7 print(A)
8 # calculate the mean of each column
9 M = mean(A.T, axis=1)
10 print(M)
11 # center columns by subtracting column means
12 C = A - M
13 print(C)
14 # calculate covariance matrix of centered matrix
15 V = cov(C.T)
16 print(V)
17 # eigendecomposition of covariance matrix
18 values, vectors = eig(V)
19 print(vectors)
20 print(values)
21 # project data
22 P = vectors.T.dot(C.T)
23 print(P.T)
```

Fuente:

<https://machinelearningmastery.com/calculate-principal-component-analysis-scratch-python/>

# Comentarios

- PCA sirve para visualizar datos multidimensionales al proyectar a dos dimensiones.
- PCA ayuda a métodos basados en distancias (KNN, K-means) que sufren con la maldición de la dimensionalidad
- Nuestros nuevos atributos pierden interpretabilidad.
- Complejidad: Para  $n$  atributos y  $m$  datos, calcular la matriz de covarianza cuesta  $O(n^2m)$ , la descomposición en vectores propios es  $O(n^3)$ .
- Complejidad es  $O(n^2m+n^3)$ .
- Los componentes principales también se pueden obtener aplicando descomposición por valores singulares SVD sobre la matriz de datos ( en vez de usar la matriz de covarianza)
- SVD tiende a ser más estable.

# PCA: Ejemplo EigenFaces

input: dataset of  $N$  face images

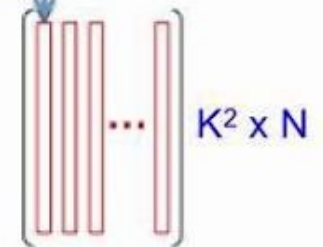


face:  $K \times K$  bitmap of pixels



“unfold” each bitmap to  $K^2$ -dimensional vector

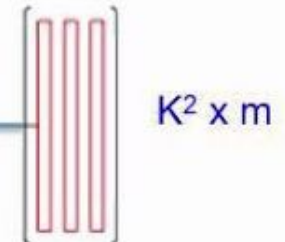
arrange in a matrix  
each face = column



“fold” into a  $K \times K$  bitmap



PCA



set of  $m$  eigenvectors  
each is  $K^2$ -dimensional

# Multidimensional Scaling

- MDS es una técnica para descubrir la estructura espacial subyacente de un conjunto de datos
  - Basado en las disimilitudes entre los objetos
- Se utiliza para
  - Reducción de atributos
  - Mapear objetos complejos a espacio vectorial



# Multidimensional Scaling

- Algoritmo MDS
  - Entrada:
    - Conjunto de  $m$  objetos de  $n$  dimensiones
    - Matriz de distancias
    - Dimensión  $k$  objetivo ( $k \ll n$ )
  - Algoritmo mapea cada objeto a un punto  $k$ -D, minimizando una función de stress
    - La función de stress mide el error relativo de las distancias en el espacio objetivo

# Multidimensional Scaling

- Algoritmo MDS
  - Función de stress:

$$stress = \sqrt{\left( \frac{\sum_{i,j} (\hat{d}_{ij} - d_{ij})^2}{\sum_{i,j} d_{ij}^2} \right)}$$

$d_{ij}$ : distancia entre objetos  $O_i$  y  $O_j$ .

$\hat{d}_{ij}$ : distancia Euclidiana entre sus imgenes  $P_i$  y  $P_j$ .

# Multidimensional Scaling

- Algoritmo MDS:
  - MDS comienza con un conjunto de vectores  $k$ -dimensionales iniciales (e.g., aleatorios)
  - Luego, itera modificando los vectores de forma de reducir el stress
- Diversas formas de implementar MDS
  - Algoritmo SMACOF (algoritmo iterativo que minimiza la función de stress)
- Se puede aplicar con distancias no-métricas
- Complejidad temporal:  $O(m^2)$

# Otras técnicas

- Autoencoders
  - Red neuronal que reconstruye los datos. Se usa la capa intermedia como una representación.
- t-Distributed Stochastic Neighbor Embedding (TSNE).
  - Modela cada objeto de alta dimensión por un punto de dos o tres dimensiones de tal manera que los objetos similares son modelados por puntos cercanos y los objetos diferentes son modelados por puntos distantes con alta probabilidad.
  - ¡Técnica muy buena para visualización!
  - Explicado en este video: <https://www.youtube.com/watch?v=NEaUSP4YerM>
- ICA (Independent Component Analysis)
- UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.
  - Basado geometría de Riemann y topología algebraica: <https://www.youtube.com/watch?v=nq6iPZVUxZU>



**dcc**

CIENCIAS DE LA COMPUTACIÓN  
UNIVERSIDAD DE CHILE

[www.dcc.uchile.cl](http://www.dcc.uchile.cl)

f @ in  / DCCUCHILE