Aprendizaje Automático

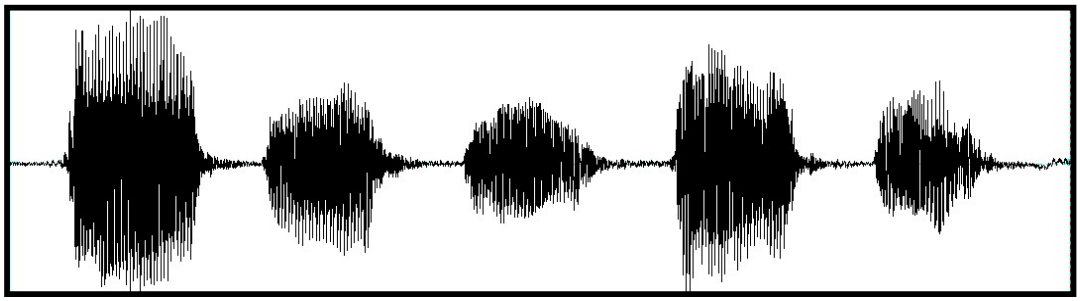
Segundo Cuatrimestre de 2016

Reducción de Dimensionalidad

DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Ejemplo: Procesamiento del Habla



Señal de audio cruda:
16bits @ 8-16kHz

- La señal se corta en segmentos de ancho variable (desde 10ms hasta varios segundos).
- Pueden extraerse cientos de atributos de cada segmento.
  - Ejemplo: OpenSmile.
  - (34 low-level descriptors (pcm_loudness, mfcc, logMelFreqBand, f0finEnv, etc.) + 34 delta coefficients) x 21 functionals (max, min, quartiles, mean, etc.) + ……..

Tantos atributos pueden causar problemas...

- Eficiencia del entrenamiento:
  - Espacio de hipótesis crece con $|Atributos|$.
  - Ej: árboles de decisión: $O(|Instancias| \cdot |Atributos|^2)$
- Calidad del modelo entrenado:
  - Por atributos no independientes (ej: NB);
  - Por atributos sin información (ej: kNN).
- Costo de extracción:
  - Algoritmos no triviales de extracción.
- Costo de almacenamiento y transmisión:
  - Ej: para Reconocimiento del Habla, se usan vectores de 40 atributos extraídos cada 10ms (4000 atr/seg).

Otro ejemplo

- Reconocimiento de caras

- Cara: NxM pixels (ej: 640 x 480 ~ 300k atributos)
- Las caras humanas son todas parecidas, con pequeñas variaciones que las distinguen.
- Desafío: codificar eficientemente esas variaciones.

http://home.iitk.ac.in/~draj/cs676/project/paper.html

Selección de Atributos

- Dados:

  F: conjunto de atributos

  L: algoritmo de aprendizaje

  D: datos de entrenamiento

  Perf: medida del desempeño (*accuracy*, *F-measure*, etc.)

- Buscamos la mejor selección de atributos *S*:

$$\underset{S \subseteq F}{\operatorname{argmax}} \; Perf(S, L, D)$$

Búsqueda Exhaustiva

1. **Para cada** $S \in \text{Partes}(F)$:

   Computar $Perf(S, L, D)$

2. **Retornar** S con mejor medida de desempeño $Perf$

- Encuentra la mejor solución, pero en tiempo exponencial en $|F|$.
- Debemos usar técnicas aproximadas de búsqueda.
  - AyED3, Metaheurísticas, etc.

Ranking de Atributos

1. Elegir una métrica para evaluar un atributo *A*:

   Ejemplos: $Ganancia(D, A) = H(D) - \sum_{v \in Valores(A)} \frac{|D_v|}{|D|} H(D_v)$

   $GainRatio(D, A)$

2. Evaluar cada atributo individualmente.
3. Generar un ranking.
4. Quedarse con los *k* mejores atributos.

Problemas:

- Atributos parecidos pueden monopolizar los *top-k*.
- No captura interacciones entre atributos.

Greedy Forward Selection

- Partiendo del conjunto vacío, agregar de a uno los atributos que más mejoren el desempeño.

1. $S \leftarrow \emptyset$

2. $f_{mejor} \leftarrow \underset{f \in F}{\operatorname{argmax}}\ Perf(S \cup \{f\}, L, D)$

3. **Si** $Perf(S \cup \{f_{mejor}\}, L, D) > Perf(S, L, D)$ **:**

- **i.** $S \leftarrow S \cup \{f_{mejor}\}$
- **ii.** $F \leftarrow F \setminus \{f_{mejor}\}$
- **iii.** **Si** $\mathrm{F} \neq \emptyset$ **: Volver al paso 2.**

4. **Retornar** S

Greedy Backward Elimination

- Partiendo del conjunto con todos los atributos, borrar de a uno los atributos que más empeoren el desempeño.

1. $S \leftarrow \text{copia de } F$

2. $f_{peor} \leftarrow \underset{f \in S}{\text{argmax}}\ Perf(S \setminus \{f\}, L, D)$

3. **Si** $Perf(S \setminus \{f_{peor}\}, L, D) > Perf(S, L, D)$ **:**

$S \leftarrow S \setminus \{f_{peor}\}$

Volver al paso 2.

4. **Retornar** S

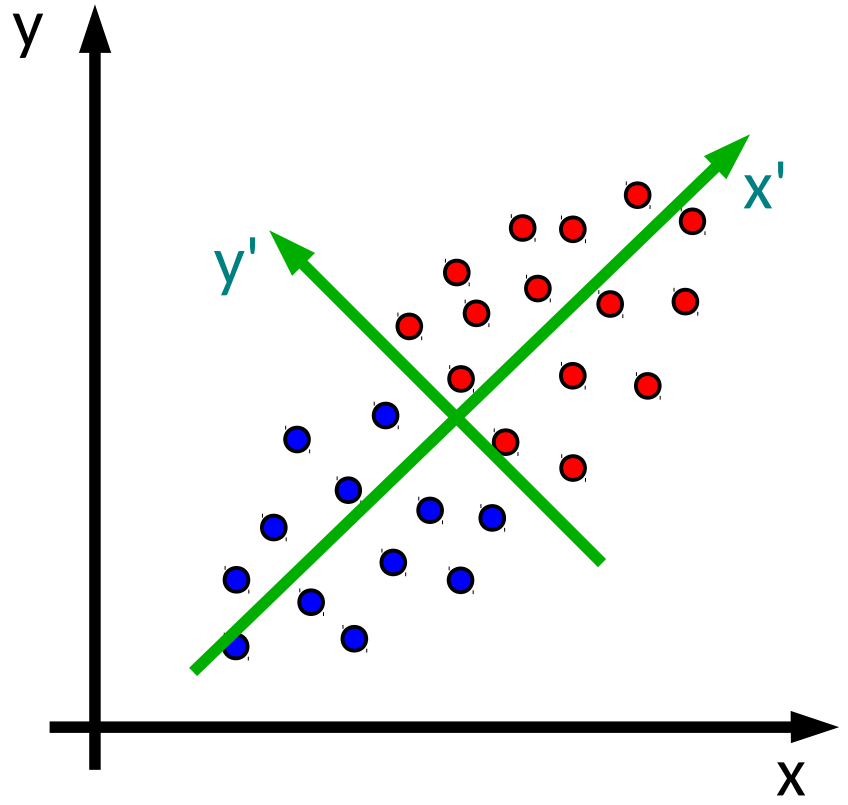
Búsqueda Hill-Climbing (Best-First)

- Comenzar con $S \subseteq F$ elegido al azar.
- Definimos a los vecinos de S como:
  - $S \cup \{f\}$ para algún $f \notin S$
  - $S \setminus \{f\}$ para algún $f \in S$
- En cada iteración, buscar el vecino de S con mejor desempeño.
- Terminar cuando no se pueda mejorar.

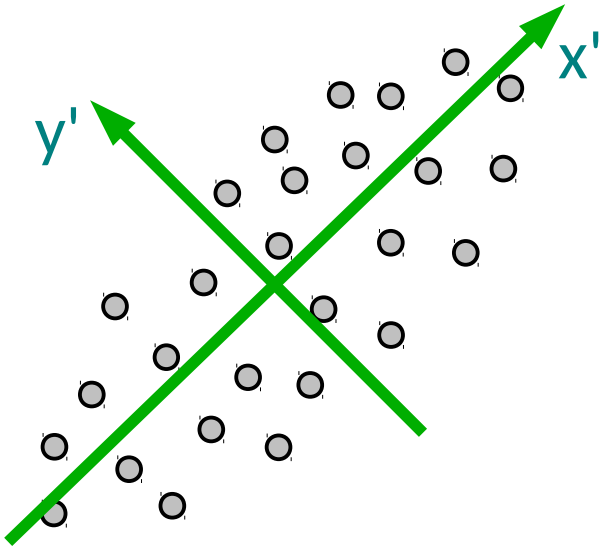
Transformación de Atributos

- Reconocimiento de caras
- Cara: NxM pixels (ej: 640 x 480 ~ 300k atributos)
- Queremos codificar eficientemente las variaciones de las diferentes caras.
  - Selección de atributos: no ayuda.
  - Principal Component Analysis (PCA).

Principal Component Analysis



Principal Component Analysis



- En el ejemplo anterior mostramos puntos rojos y azules, pero la elección de x' e y' se hace a ciegas de las clases de las instancias.
- PCA es independiente de las clases de las instancias.
- Por eso, sirve también para aprendizaje no supervisado.

Principal Component Analysis

- Datos: Observaciones $\{\mathbf{x}_n\}$ donde $n = 1, \ldots, N$

  Cada $\mathbf{x}_n$ es un vector de D dimensiones.

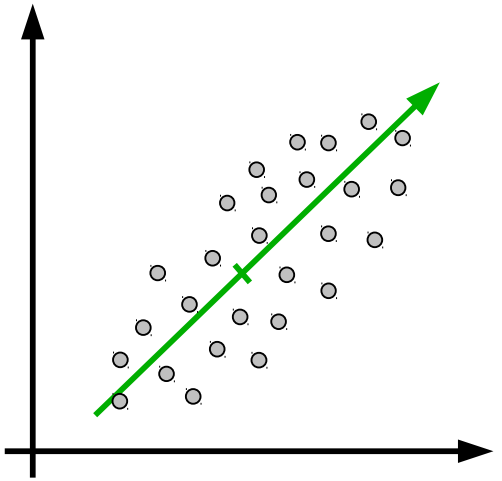
- Objetivo:

  a) Proyectar los datos a un espacio de $M < D$ dimensiones,

  b) Maximizando la varianza de los datos proyectados.

  o bien, equivalentemente:

  b') Minimizando el error cuadrático medio de las proyecciones.



Principal Component Analysis
(Bishop, capítulo 12)

- Ambas definiciones de PCA llevan a la misma equación:

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

donde **S** es la matriz de covarianza:

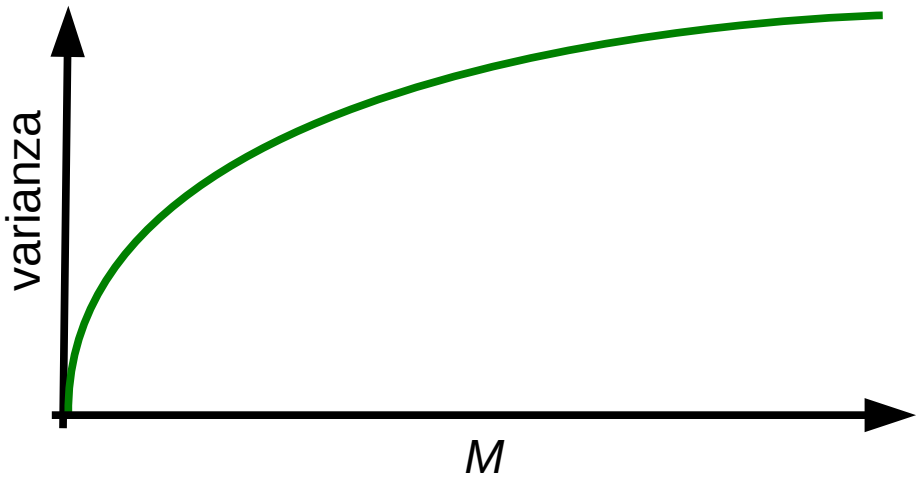
$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^{\mathrm{T}}$$

- $\mathbf{u}_i$: autovectores de S. Componentes del nuevo espacio.
- λ_i : autovalores de S. Varianza de los datos en la dirección $\mathbf{u}_i$.
- PCA: Ordenar las nuevas componentes $\mathbf{u}_i$ según λ_i.

Principal Component Analysis

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

- $\mathbf{u}_i$: autovectores de S. Componentes del nuevo espacio.
- λ_i : autovalores de S. Varianza de los datos en la dirección $\mathbf{u}_i$.
- PCA: Ordenar las nuevas componentes $\mathbf{u}_i$ según λ_i.
- ¿Cuántas componentes?



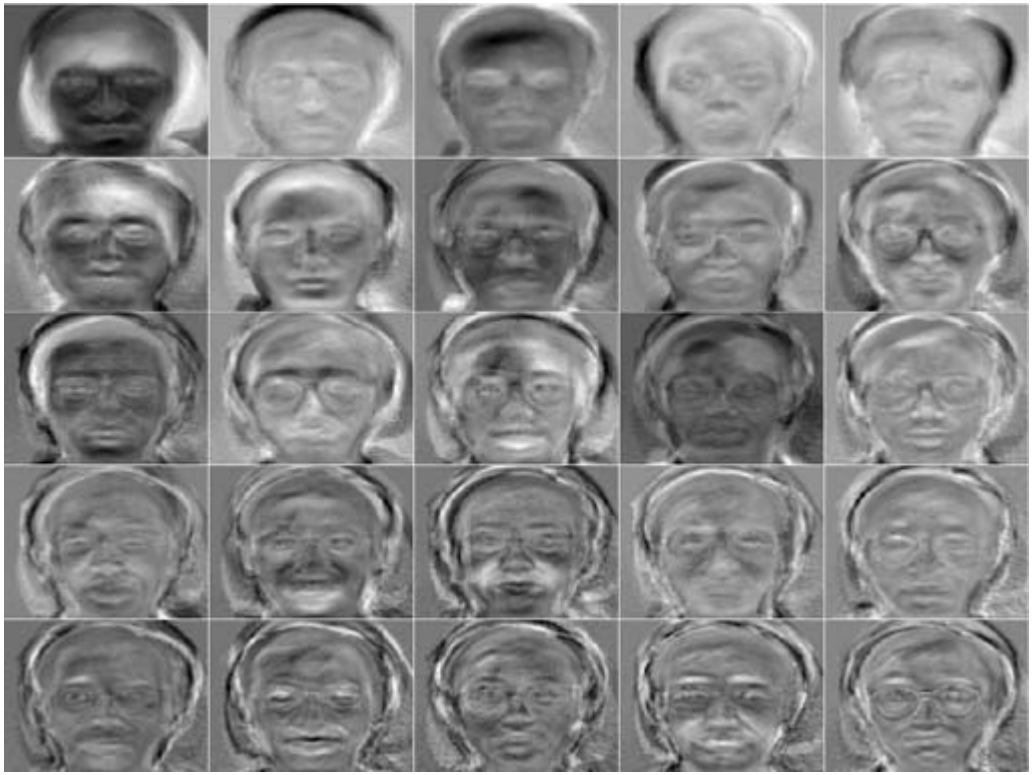
- Aplicación de PCA: Compresión de datos con pérdida (*lossy compression*).

Eigenfaces

http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Base de datos de caras:
$\{\mathbf{x}_n\}$, cada $\mathbf{x}_n$ tiene D pixels

Autovectores:
(25 principales)



Eigenfaces

http://www.cs.princeton.edu/~cdecoro/eigenfaces/

Reconstrucción
Agregando 1 componente principal cada vez.

Reconstrucción
Agregando 8 componentes principales cada vez.

Resumen

- Reducción de dimensionalidad
- Selección de atributos: ranking, greedy, etc.
- Transformación de atributos: PCA