



A background scatter plot featuring several distinct clusters of points. One cluster in the center is yellow, surrounded by green, pink, purple, and blue clusters. A red cluster is visible at the bottom. The background is a light gray.

El aprendizaje de métricas de distancia

Trabajo de Fin de Grado

Juan Luis Suárez Díaz

Tutores:

Francisco Herrera Triguero

Salvador García López

1 Introducción

- Objetivos
- Descripción del problema
- Aplicaciones

2 Matemáticas

3 Informática teórica

4 Informática práctica

5 Conclusiones y vías futuras

Objetivos

- Conocer la disciplina del aprendizaje de métricas de distancia.
- Estudiar los fundamentos matemáticos del aprendizaje de métricas de distancia.
- Analizar los principales algoritmos de aprendizaje de métricas de distancia.
- Desarrollar un software que integre los algoritmos de aprendizaje estudiados.

El aprendizaje de métricas de distancia

¿Qué es?

Es una rama del aprendizaje automático cuya finalidad es aprender distancias a partir de los datos.

Definición

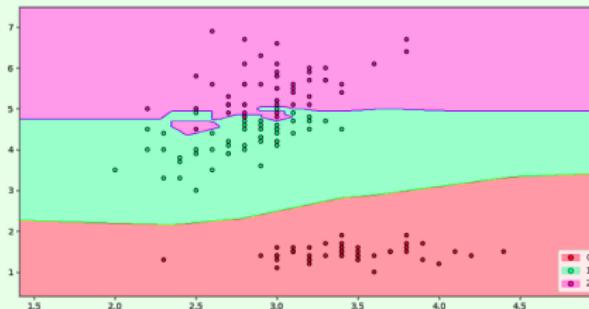
Sea X un conjunto no vacío. Una **distancia** sobre X es una aplicación $d: X \times X \rightarrow \mathbb{R}$, verificando:

- ① $d(x, y) = 0 \iff x = y$ para cualesquiera $x, y \in X$ (*coincidencia*)
 - ② $d(x, y) = d(y, x)$ para cualesquiera $x, y \in X$ (*simetría*)
 - ③ $d(x, z) \leq d(x, y) + d(y, z)$ para cualesquiera $x, y, z \in X$ (*desigualdad triangular*).
- **Pseudodistancias:** Exigen solo $d(x, x) = 0$ en ①.

¿Por qué distancias?

Ejemplo

Los clasificadores de vecinos cercanos



Problema

- Los algoritmos basados en distancias suelen utilizar distancias fijas.
 - **Solución:** Aprender distancias.

¿Cómo aprender una distancia?

Definición (*Distancias de Mahalanobis*)

Sea $M \in \mathcal{M}_d(\mathbb{R})$ semidefinida positiva. Entonces, $d_M : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, dada por

$$d(x, y) = \sqrt{(x - y)^T M (x - y)}$$

es una (pseudo-)distancia, denominada **distancia de Mahalanobis**.

Enfoque principal del aprendizaje de métricas de distancia

Aprender distancias de Mahalanobis sobre espacios vectoriales d -dimensionales.

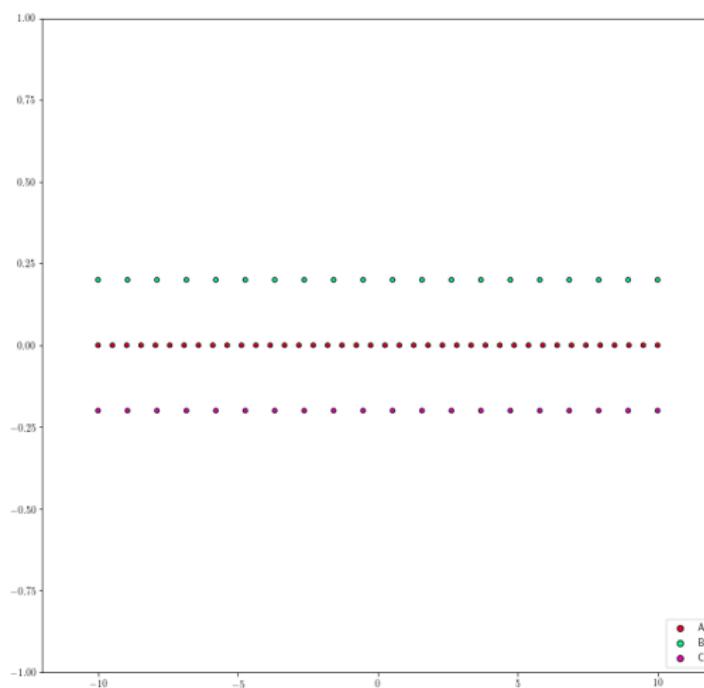
Dos opciones:

- ① Aprender M .
- ② Aprender una aplicación lineal L .

Entonces, $M = L^T L$ y $d_M(x, y)^2 = \|L(x - y)\|_2^2$.

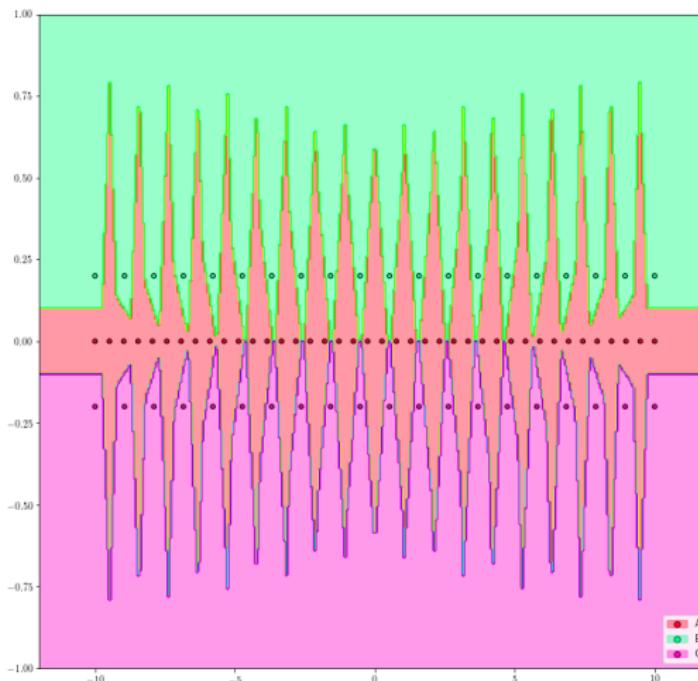
Mejora de clasificadores basados en distancias

1-NN



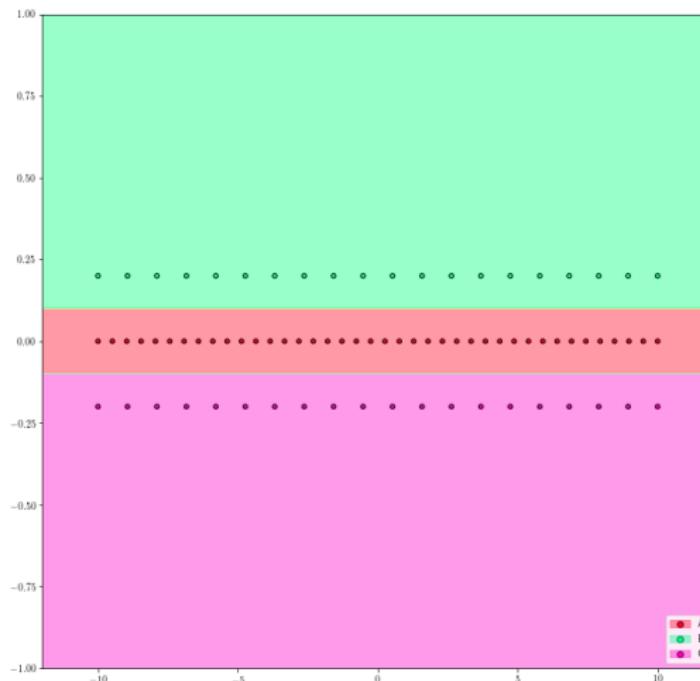
Mejora de clasificadores basados en distancias

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



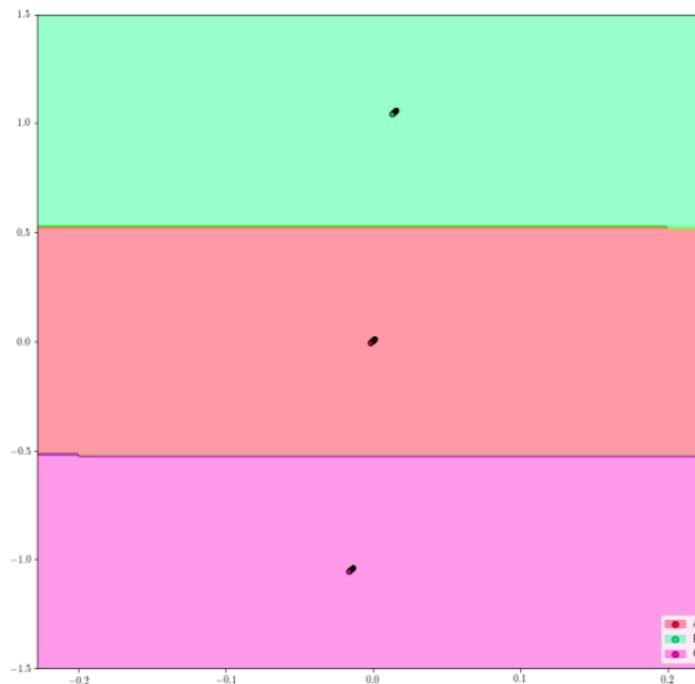
Mejora de clasificadores basados en distancias

$$M \approx \begin{pmatrix} 0 & -0.004 \\ -0.004 & 27.5 \end{pmatrix}$$



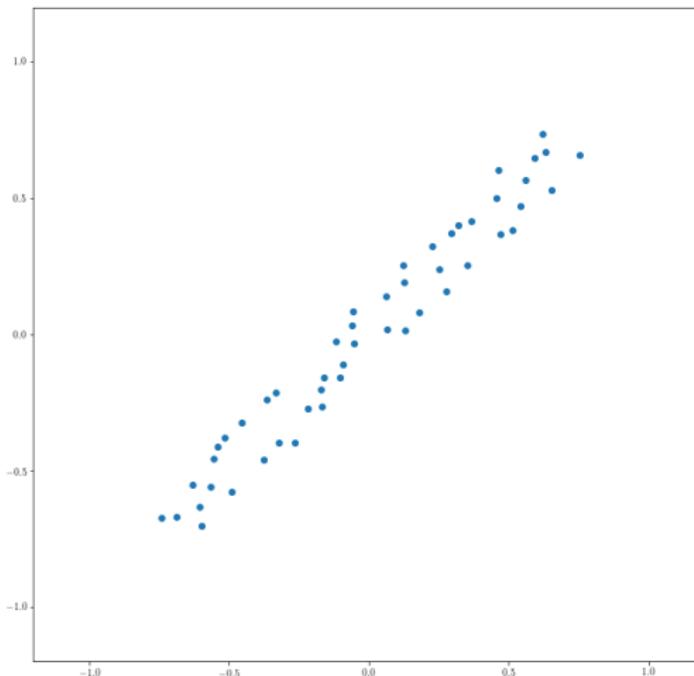
Mejora de clasificadores basados en distancias

$$L \approx \begin{pmatrix} -0.0001 & 0.073 \\ -0.0008 & 5.24 \end{pmatrix}$$



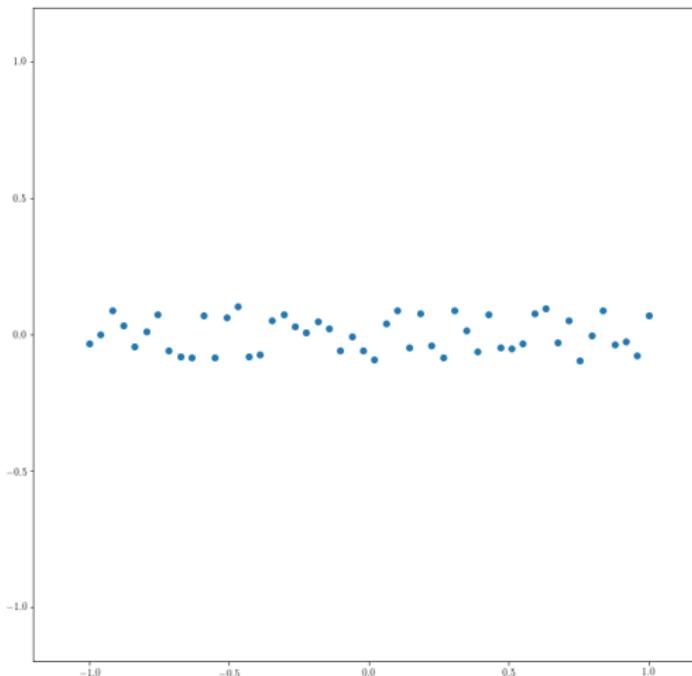
Organización de datos y reducción de dimensionalidad

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



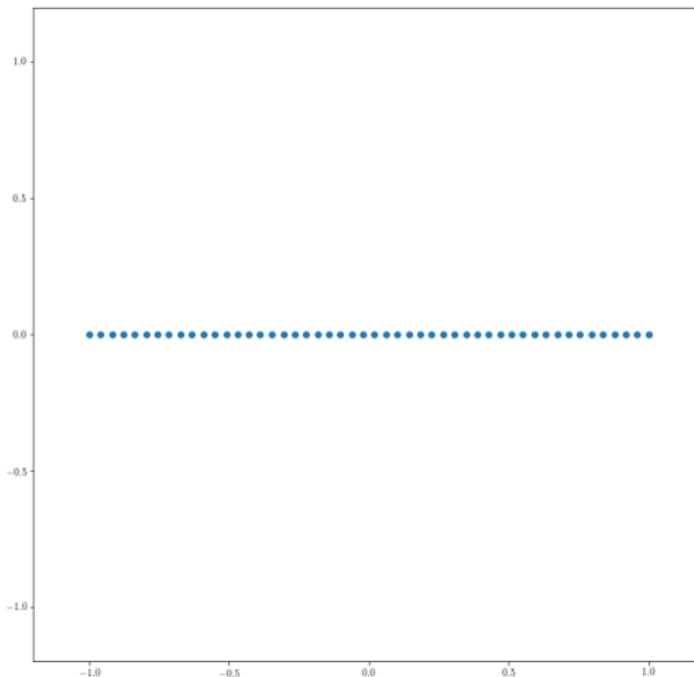
Organización de datos y reducción de dimensionalidad

$$L = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix}$$



Organización de datos y reducción de dimensionalidad

$$L = (\sqrt{2}/2 \quad \sqrt{2}/2)$$



1 Introducción

2 Matemáticas

- Análisis convexo
- Análisis matricial
- Teoría de la información

3 Informática teórica

4 Informática práctica

5 Conclusiones y vías futuras

Las matemáticas bajo el aprendizaje de métricas de distancia

- ① **Análisis convexo.** De gran importancia en la mayoría de algoritmos de aprendizaje de métricas de distancia.
- ② **Análisis matricial.** Las matrices son la herramienta fundamental para modelar el problema.
- ③ **Teoría de la información.** Presente en algunos de los algoritmos.

Hiperplanos soporte

Definición (*Hiperplano soporte*)

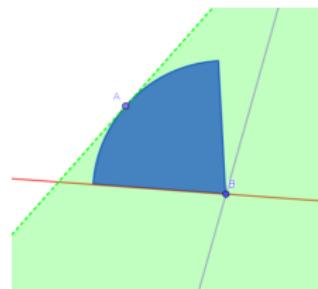
Sean $T: \mathbb{R}^d \rightarrow \mathbb{R}$ lineal, $\alpha \in \mathbb{R}$ y $P = \{x \in \mathbb{R}^d : T(x) = \alpha\}$ hiperplano.

Definimos $P^+ = \{x \in \mathbb{R}^d : T(x) \geq \alpha\}$ y $P^- = \{x \in \mathbb{R}^d : T(x) \leq \alpha\}$.

P es un **hiperplano soporte** para $K \subset \mathbb{R}^d$ si $P \cap \overline{K} \neq \emptyset$ y $K \subset P^+$ o $K \subset P^-$.

Teorema (*Teorema del hiperplano soporte*)

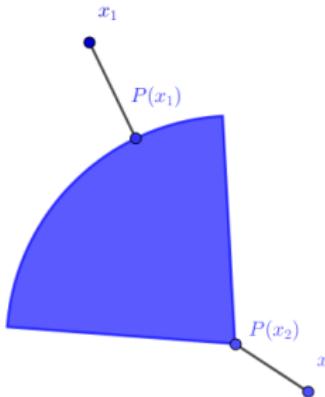
- ① Si $K \subset \mathbb{R}^d$ es convexo y cerrado, para cada $x_0 \in \text{Fr } K$ existe un hiperplano soporte P de K tal que $x_0 \in P$.
- ② Todo conjunto convexo cerrado y propio de \mathbb{R}^d es la intersección de todos sus semiespacios soporte.
- ③ Sea $K \subset \mathbb{R}^d$ un conjunto cerrado con interior no vacío. Entonces, K es convexo si y solo si para todo $x \in \text{Fr } K$ existe un hiperplano soporte P de K con $x \in P$.



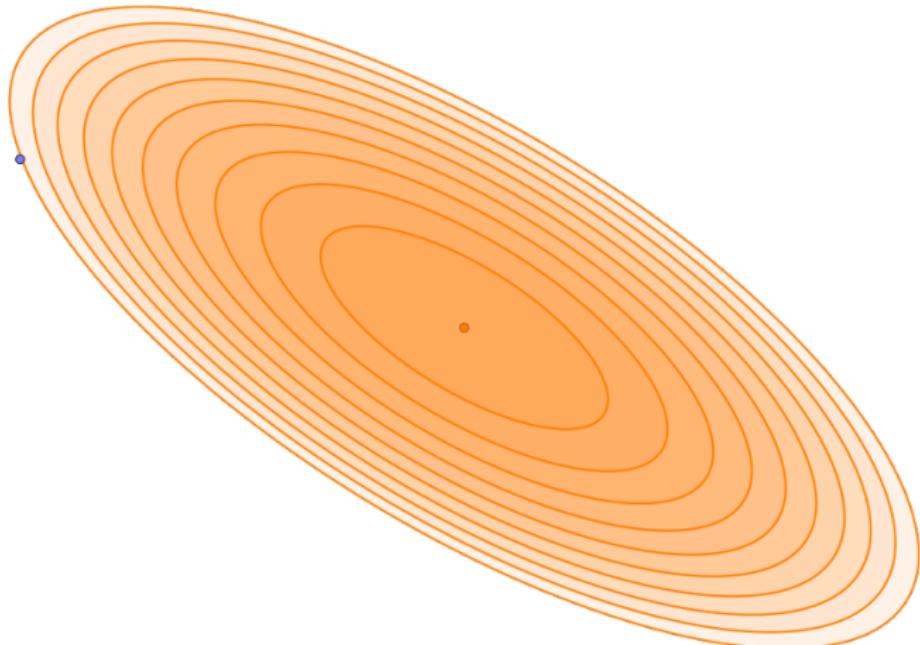
Proyecciones convexas

Teorema (*Proyección convexa*)

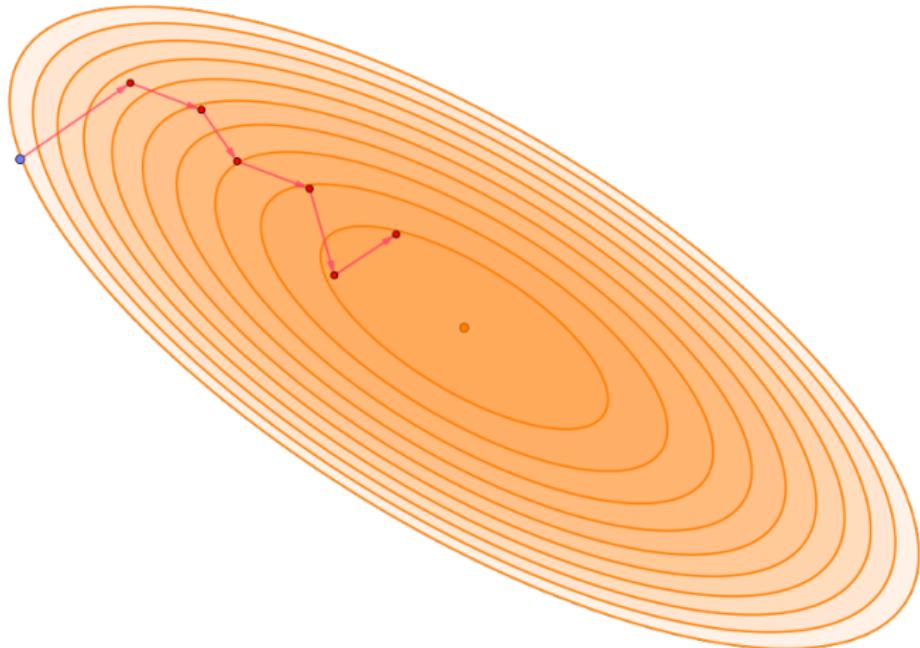
Si $K \subset \mathbb{R}^d$ es no vacío, cerrado y convexo, entonces, para cada $x \in \mathbb{R}^d$ existe un único punto $P_K(x) \in K$ tal que $d(x, K) = d(x, P_K(x))$: la proyección convexa de x sobre K .



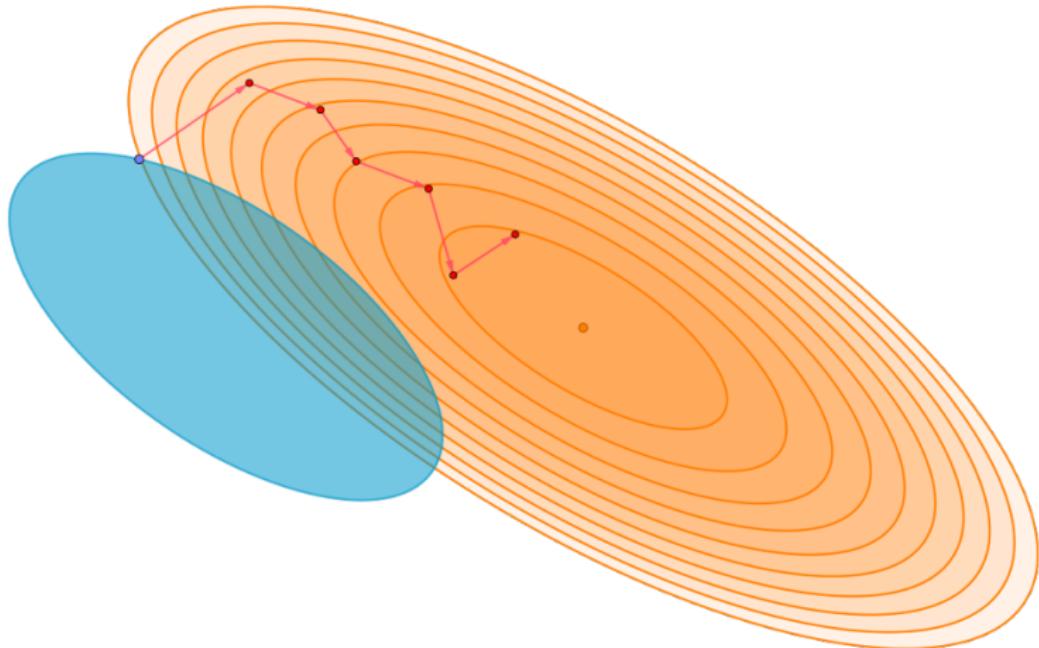
Métodos de optimización: gradiente descendente y gradiente con proyecciones



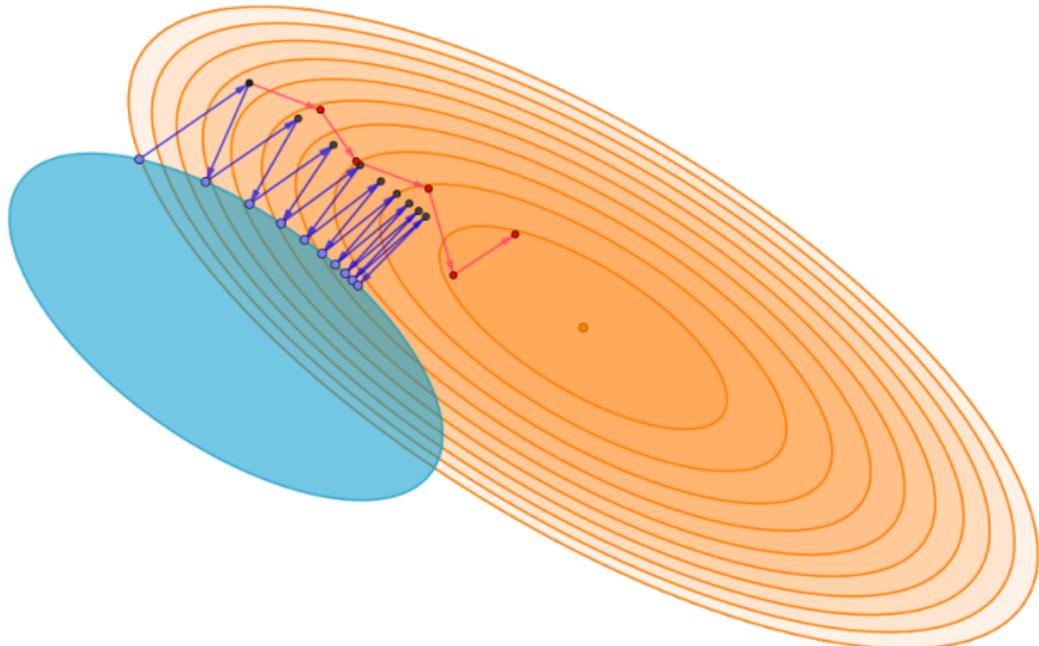
Métodos de optimización: gradiente descendente y gradiente con proyecciones



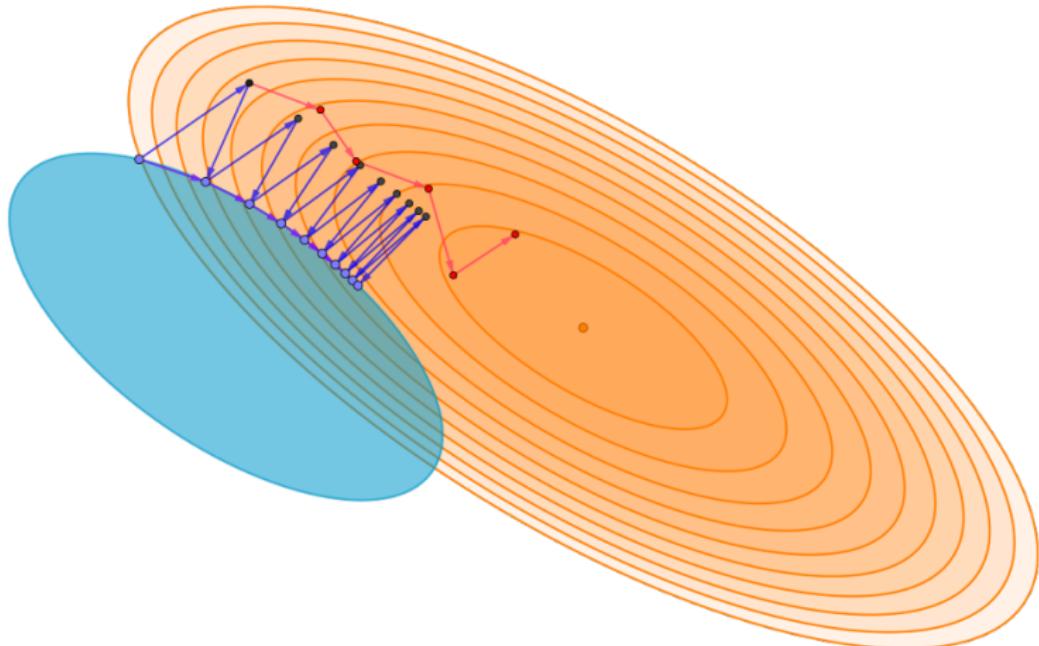
Métodos de optimización: gradiente descendente y gradiente con proyecciones



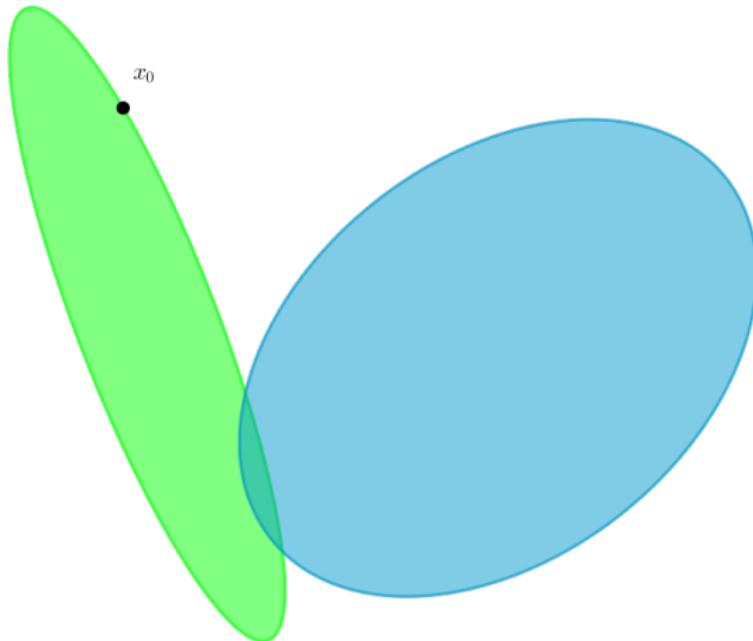
Métodos de optimización: gradiente descendente y gradiente con proyecciones



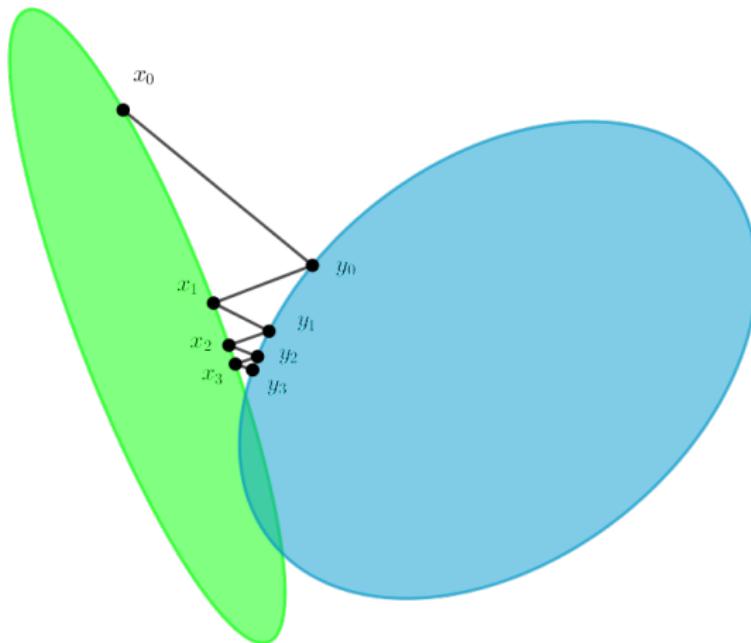
Métodos de optimización: gradiente descendente y gradiente con proyecciones



Método de las proyecciones iteradas



Método de las proyecciones iteradas



Matrices como espacio de Hilbert

Definición (*Producto escalar y norma de frobenius*)

$$\langle A, B \rangle_F = \sum_{i=1}^{d'} \sum_{j=1}^d A_{ij} B_{ij} = \text{tr}(A^T B)$$

$$\|A\|_F = \sqrt{\langle A, A \rangle_F} = \sqrt{\sum_{i=1}^{d'} \sum_{j=1}^d A_{ij}^2} = \sqrt{\text{tr}(A^T A)}$$

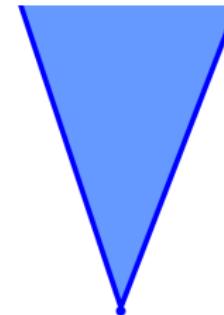
$$(\mathcal{M}_{d' \times d}(\mathbb{R}), \langle \cdot, \cdot \rangle_F) \quad \longleftrightarrow \quad \left(\mathbb{R}^{d' \times d}, \langle \cdot, \cdot \rangle \right)$$

$$\begin{pmatrix} a_{11} & \dots & a_{1d} \\ \vdots & & \vdots \\ a_{d'1} & \dots & a_{d'd} \end{pmatrix} \quad \longleftrightarrow \quad (a_{11}, \dots, a_{1d}, \dots, a_{d'1}, \dots, a_{d'd})$$

Matrices semidefinidas positivas como cono

Definición (Cono)

- Un **cono** es un conjunto C cerrado para combinaciones lineales con coeficientes no negativos.
- C es **sólido** si $C^\circ \neq \emptyset$.
- C es **puntiagudo** si $C \cap (-C) = \{0\}$.
- C es **propio** si es cerrado, sólido y puntiagudo.



El cono de las matrices semidefinidas positivas ($\mathcal{M}_d(\mathbb{R})_0^+$)

Las matrices semidefinidas positivas forman un cono propio sobre el espacio de matrices simétricas, cuyo interior es $\mathcal{M}_d(\mathbb{R})^+$.

Orden parcial de matrices como cono propio

$$A, B \in S_d(\mathbb{R}), \quad A \preceq B \iff B - A \in \mathcal{M}_d(\mathbb{R})_0^+$$

$$(\mathcal{M}_d(\mathbb{R})_0^+ \subset S_d(\mathbb{R}), \preceq) \iff (\mathbb{R}_0^+ \subset \mathbb{R}, \leq)$$

Teoremas de descomposición

Conceptos

- **Raíz cuadrada:** Dada $M \in \mathcal{M}_d(\mathbb{R})_0^+$,
 $\exists^1 N \in \mathcal{M}_d(\mathbb{R})_0^+ : N^2 = M \quad (M^{1/2} := N)$
- **Módulo:** Dada $A \in \mathcal{M}_d(\mathbb{R})$, $|A| = (A^T A)^{1/2} \in \mathcal{M}_d(\mathbb{R})_0^+$.
- **Valores singulares:** Dada $A \in \mathcal{M}_d(\mathbb{R})$, sus valores singulares son los valores propios de $|A|$.

Resultados

- **Descomposición polar:** $A \in \mathcal{M}_d(\mathbb{R}) \implies \exists U \in O_d(\mathbb{R}) : A = U|A|$.
- **Descomposición en valores singulares:**
 $A \in \mathcal{M}_d(\mathbb{R}) \implies \exists V, W \in O_d(\mathbb{R}) : A = W\Sigma V^T$, donde Σ es diagonal y contiene los valores singulares de A .
- **Descomposición $L^T L$:**
 - ① $M \in \mathcal{M}_d(\mathbb{R})_0^+ \iff M = L^T L$, con $L \in \mathcal{M}_d(\mathbb{R})$.
 - ② En tal caso, si $M = K^T K$, entonces $K = UL$, con $U \in O_d(\mathbb{R})$ (L es única salvo isometrías).

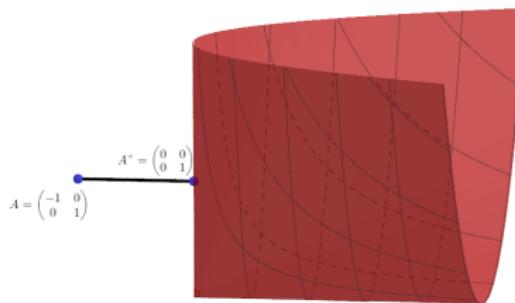
Proyección semidefinida

Definición (*Parte positiva*)

- ① Si $D = \text{diag}(\lambda_1, \dots, \lambda_d)$, $D^+ = \text{diag}(\max\{\lambda_1, 0\}, \dots, \max\{\lambda_d, 0\})$.
- ② Si $A \in S_d(\mathbb{R})$ y $A = UDU^T$, $A^+ = U D^+ U^T$.

Teorema (*Proyección semidefinida*)

Si $A \in S_d(\mathbb{R})$, A^+ es la proyección de A sobre $\mathcal{M}_d(\mathbb{R})_0^+$.



Cociente de Rayleigh

Definición (*Cociente de Rayleigh*)

$A \in S_d(\mathbb{R}), \rho_A: \mathbb{R}^d \setminus \{0\} \rightarrow \mathbb{R}$

$$\rho_A(x) = \frac{x^T A x}{x^T x}$$

Definición (*Cociente de Rayleigh generalizado*)

$A \in S_d(\mathbb{R}), B \in \mathcal{M}_d(\mathbb{R})^+, \mathcal{R}_{A,B}: \mathbb{R}^d \setminus \{0\} \rightarrow \mathbb{R}$

$$\mathcal{R}_{A,B}(x) = \frac{x^T A x}{x^T B x}$$

Optimización con vectores propios

Teorema

Sean $d', d \in \mathbb{N}$, con $d' \leq d$. Sean $A \in S_d(\mathbb{R})$, y consideramos el problema de optimización

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \text{tr}(LAL^T)$$

Entonces, el problema alcanza un
máximo si $L = \begin{pmatrix} - & v_1 & - \\ & \dots & \\ - & v_{d'} & - \end{pmatrix}$, donde
 $v_1, \dots, v_{d'}$ son vectores propios
ortonormales de A correspondientes a
sus d' mayores valores propios.

Teorema

Sean $d', d \in \mathbb{N}$, con $d' \leq d$. Sean $A \in S_d(\mathbb{R})$ y $B \in \mathcal{M}_d(\mathbb{R})^+$, y consideramos el problema de optimización

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \text{tr} \left((LBL^T)^{-1} (LAL^T) \right)$$

Entonces, el problema alcanza un máximo si $L = \begin{pmatrix} -v_1 & - \\ \dots & \\ -v_{d'} & - \end{pmatrix}$, donde $v_1, \dots, v_{d'}$ son los vectores propios de $B^{-1}A$ correspondientes a sus d' mayores valores propios.

Divergencias

Definición (*Divergencia*)

Sea X un conjunto no vacío. Una **divergencia** es una aplicación $D(\cdot\|\cdot): X \times X \rightarrow \mathbb{R}$ verificando:

- ① $D(x\|y) \geq 0$ para todos $x, y \in X$ (no negatividad).
- ② $D(x\|y) = 0$ si y solo si $x = y$.

Definición (*Divergencia de Kullback-Leibler*)

$$\text{KL}(p\|q) = \mathbb{E}_p \left[\log \frac{p(X)}{q(X)} \right]$$

Definición (*Divergencia de Jeffrey*)

$$\text{JF}(p\|q) = \text{KL}(p\|q) + \text{KL}(q\|p)$$

Algunos resultados

Teorema (*Desigualdad de la información*)

KL es una divergencia.

Definición (*Divergencia log-det*)

Sean $A, B \in \mathcal{M}_d(\mathbb{R})^+$.

$$D_{ld}(A\|B) = \text{tr}\left(AB^{-1}\right) - \log \det\left(AB^{-1}\right) - d.$$

Teorema

- ① D_{ld} es una divergencia matricial.
- ② Si $p_1 \sim \mathcal{N}(x|\mu_1, \Sigma_1)$ y $p_2 \sim \mathcal{N}(x|\mu_2, \Sigma_2)$, entonces

$$\text{KL}(p_1\|p_2) = \frac{1}{2}D_{ld}(\Sigma_1\|\Sigma_2) + \frac{1}{2}d_{\Sigma_2^{-1}}^2(\mu_1, \mu_2).$$

Introducción
oooooo

Matemáticas
oooooooooooo

Informática teórica
ooooooo

Informática práctica
oooooooooooo

Conclusiones y vías futuras
oo

1 Introducción

2 Matemáticas

3 Informática teórica

- El aprendizaje de métricas de distancia
- Análisis de algoritmos

4 Informática práctica

5 Conclusiones y vías futuras

Aprendizaje de métricas de distancia

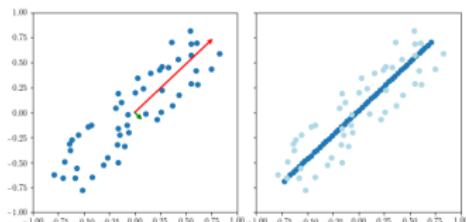
- Aprender distancias a partir de los datos.
- Dos enfoques: aprender M o aprender L .
- Mejoran algoritmos de aprendizaje por semejanza y pueden reducir dimensionalidad.

Algoritmos supervisados: estrategia

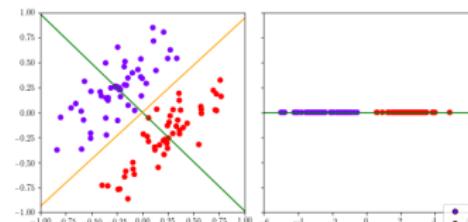
$$\min_{M \in \mathcal{M}_d(\mathbb{R})_0^+} \ell(d_M, (x_1, y_1), \dots, (x_N, y_N))$$

Algoritmos centrados en reducción de dimensionalidad

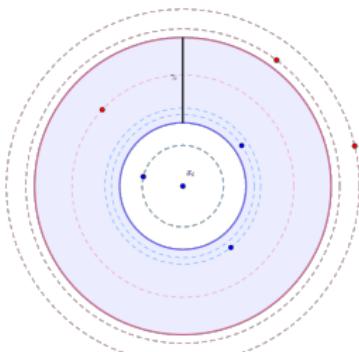
PCA



LDA



ANMM



- **PCA:**

$$\max_{\substack{L \in \mathcal{M}_{d' \times d}(\mathbb{R}) \\ LL^T = I}} \text{tr}(L\Sigma L^T),$$

- **LDA:**

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \text{tr}((LS_wL^T)^{-1}(LS_bL^T)).$$

- **ANMM:**

$$\max_{\substack{L \in \mathcal{M}_{d' \times d}(\mathbb{R}) \\ LL^T = I}} \text{tr}(L(S - C)L^T),$$

Fei Wang y Changshui Zhang. "Feature extraction by maximizing the average neighborhood margin". En: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on. IEEE. 2007, págs. 1-8.

Algoritmos orientados a mejorar el clasificador de vecinos cercanos

LMNN

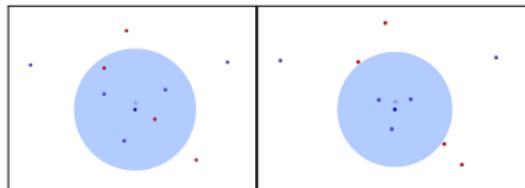
NCA

Conceptos:

- **Target neighbors:** Vecinos que queremos considerar en la clasificación.
- **Impostores:** Invaden el perímetro determinado por los target neighbors.

Objetivos:

- Acercar los target neighbors lo máximo posible.
- Eliminar los impostores.



Kilian Q Weinberger y Lawrence K Saul. "Distance metric learning for large margin nearest neighbor classification". En: Journal of Machine Learning Research 10.Feb (2009), págs. 207-244.

- Enfoque probabilístico.
- Maximización de la tasa de acierto esperada por la clasificación 1-NN.
- Probabilidad de que x_i tenga a x_j como su vecino más cercano:

$$p_{ij}^L = \frac{\exp(-\|Lx_i - Lx_j\|^2)}{\sum_{k \neq i} \exp(-\|Lx_i - Lx_k\|^2)} \quad (j \neq i),$$
$$p_{ii}^L = 0$$

- Enfoque probabilístico
 - ⇒ Objetivo diferenciable
 - ⇒ Métodos de gradiente

Jacob Goldberger y col. "Neighbourhood components analysis". En: Advances in neural information processing systems. 2005, págs. 513-520.

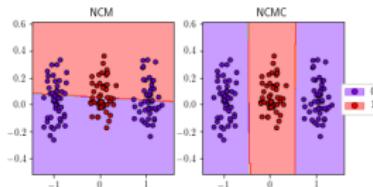
Algoritmos orientados a mejorar los clasificadores de centroides cercanos

Clasificadores basados en centroides

Establecen centroides en los datos y clasifican según el centroide más cercano.

Tipos

- ① **NCM.** Utiliza las medias de cada clase (*Nearest Class Mean*).
- ② **NMC.** Utiliza múltiples centroides por clase, establecidos mediante K-Means.



Algoritmos de Aprendizaje

- NCMLL, para NCM.
- NMC, para el clasificador NMC.
- Enfoques probabilísticos.

Thomas Mensink y col. "Metric learning for large scale image classification: Generalizing to new classes at near-zero cost". En: Computer Vision–ECCV 2012. Springer, 2012, págs. 488-501.

Algoritmos basados en teoría de la información

ITML

DMLMJ

MCML

- **Objetivo:** aproximar una métrica inicial satisfaciendo restricciones de similaridad.

- **Métrica inicial:**
 $M_0 \rightarrow p_0 \sim \mathcal{N}(x|\mu, M_0)$.

- **Variable:**
 $M \rightarrow p_M \sim \mathcal{N}(x|\mu, M)$.

- **Problema:**

$$\min_{M \succeq 0} \text{KL}(p_0 \| p_M)$$

s.a.: $d_M(x_i, x_j) \leq u, \quad (i, j) \in S$
 $d_M(x_i, x_j) \geq l, \quad (i, j) \in D$.

 Jason V Davis y col.
“Information-theoretic metric learning”. En: Proceedings of the 24th international conference on Machine learning. ACM. 2007, págs. 209-216.

- **Objetivo:** Separar lo máximo posible las distribuciones asociadas a los puntos similares y a los puntos no similares.

- **Vecindarios:** Vecinos más cercanos de igual clase ($V_k^+(x_i) \rightarrow \Sigma_S$) y distinta clase ($V_k^-(x_i) \rightarrow \Sigma_D$).

- **Distribuciones de diferencias:**

$$P_L \sim \mathcal{N}(x|0, L\Sigma_S L^T)$$

$$Q_L \sim \mathcal{N}(x|0, L\Sigma_D L^T).$$

- **Problema:**

$$\max_{L \in \mathcal{M}_{d' \times d}(\mathbb{R})} \text{JF}(P_L \| Q_L).$$

 Bac Nguyen, Carlos Morell y Bernard De Baets. “Supervised distance metric learning through maximization of the Jeffrey divergence”. En: Pattern Recognition 64 (2017), págs. 215-225.

- **Objetivo:** Aproximarse lo máximo posible a la *distribución ideal* donde los puntos de las mismas clases colapsan.

- **Distribución ideal:**

$$p_0(j|i) \propto \begin{cases} 1, & y_i = y_j \\ 0, & y_i \neq y_j \end{cases}$$

- **Distribución métrica:**

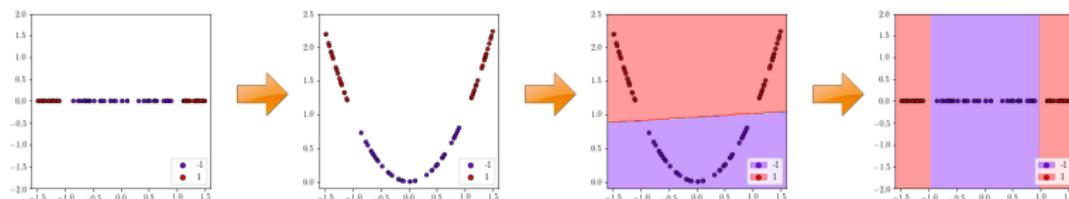
$$p^M(j|i) = \frac{\exp(-\|x_i - x_j\|_M^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|_M^2)}.$$

- **Problema:**

$$\min_{M \succeq 0} \sum_{i=1}^N \text{KL}[p_0(j|i) \| p^M(j|i)].$$

 Amir Globerson y Sam T Roweis. “Metric learning by collapsing classes”. En: Advances in neural information processing systems. 2006, págs. 451-458.

Kernel trick y algoritmos basados en kernels



Kernel Trick

- Enviar los datos a un espacio de alta dimensionalidad \mathcal{F} , mediante $\phi: \mathbb{R}^d \rightarrow \mathcal{F}$.
- Aprender una distancia en \mathcal{F} mediante $L: \mathcal{F} \rightarrow \mathbb{R}^{d'}$.
- Teoremas de representación:** $L\phi(x_i) = AK_{i,i}$, con A y K matrices calculables.
- Ventajas:** Mayor variedad de distancias.

Algoritmos kernelizados

① KLMNN

② KANMM

③ KDMLMJ

④ KDA

Otros algoritmos

LSI

$$\begin{aligned} & \underset{M}{\max} \quad \sum_{(x_i, x_j) \in D} \|x_i - x_j\|_M \\ \text{s.a.: } & \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_M^2 \leq 1 \\ & M \succeq 0. \end{aligned}$$

DML-eig

$$\begin{aligned} & \underset{M}{\max} \quad \min_{(x_i, x_j) \in D} \|x_i - x_j\|_M^2 \\ \text{s.a.: } & \sum_{(x_i, x_j) \in S} \|x_i - x_j\|_M^2 \leq 1 \\ & M \succeq 0. \end{aligned}$$

LDML

$$\begin{aligned} \sigma(x) &= \frac{1}{1 + e^{-x}}. \\ p_{ij,M} &= \sigma(b - \|x_i - x_j\|_M^2) \\ \mathcal{L}(M) &= \sum_{i,j=1}^N y_{ij} \log p_{ij,M} + (1 - y_{ij}) \log(1 - p_{ij,M}) \\ & \underset{M \succeq 0}{\max} \quad \mathcal{L}(M) \end{aligned}$$

 Eric P Xing y col. "Distance metric learning with application to clustering with side-information". En: Advances in neural information processing systems. 2003, págs. 521-528.

 Yiming Ying y Peng Li. "Distance metric learning with eigenvalue optimization". En: Journal of Machine Learning Research 13.Jan (2012), págs. 1-26.

 Matthieu Guillaumin, Jakob Verbeek y Cordelia Schmid. "Is that you? Metric learning approaches for face identification". En: Computer Vision, 2009 IEEE 12th international conference on. IEEE. 2009, págs. 498-505.

Introducción
oooooo

Matemáticas
oooooooooooo

Informática teórica
ooooooo

Informática práctica
oooooooooooo

Conclusiones y vías futuras
oo

1 Introducción

2 Matemáticas

3 Informática teórica

4 Informática práctica

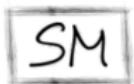
- Software desarrollado
- Experimentación

5 Conclusiones y vías futuras

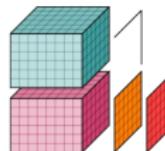
Python y R

- Lenguajes de programación de alto nivel, multiplataforma, orientados a objetos e interpretados.
- De gran popularidad en el aprendizaje automático.
- Gran cantidad de librerías para computación científica.
- Actualmente sin librerías amplias de aprendizaje de métricas de distancia.





StatsModels
Statistics in Python



xarray



scikits-image
image processing in python



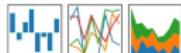
machine learning in Python



And many,
many more...



pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



NumPy



IP[y]:
IPython

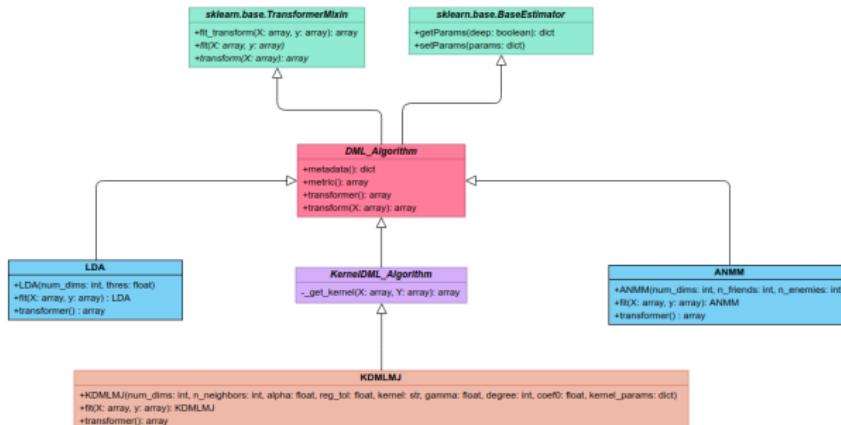


pyDML

- *Distance Metric Learning algorithms for Python.*
- Incorpora los algoritmos de aprendizaje de métricas de distancia siguiendo la estructura de la librería **Scikit-Learn**.
- Funcionalidades adicionales: clasificadores, dibujo y estimación de parámetros.



Read the Docs



① Construcción: `alg = LDA(num_dims = 2)`

② Ajuste: `alg.fit(X,y)`

③ Resultados del aprendizaje:

- Generalización: `alg.transform(newX)`
- Métrica: `M = alg.metric()`
- Aplicación lineal: `L = alg.transformer()`

Ejemplo

Aprendiendo distancias

```
>>> import numpy as np
>>> from sklearn.datasets import load_iris

>>> # Loading DM Algorithms
>>> from dml import NCA

>>> # Loading dataset
>>> iris = load_iris()
>>> X = iris['data']
>>> y = iris['target']

>>> # DML construction
>>> nca = NCA()

>>> # Fitting algorithm
>>> nca.fit(X,y)

>>> # We can look at the algorithm metadata after fitting it
>>> meta = nca.metadata()
>>> meta
{'final_expectance': 0.95771240234375,
 'initial_expectance': 0.8380401129557291,
 'num_iters': 3}

>>> # We can see the metric the algorithm has learned.
>>> # This metric is the PSD matrix that defines how the distance is measured:
>>> d(x,y) = (x-y).T.dot(M).dot(x-y)
>>> M = nca.metric()
>>> M
array([[ 1.19008678,  0.51293714, -2.15018151, -2.01464351],
       [ 0.51293714,  1.58120238, -2.14573777, -2.10714773],
       [-2.15018151, -2.14573777,  6.46861853,  5.86280474],
       [-2.01464351, -2.10714773,  5.86280474,  6.83271473]])
```

```
>>> # Equivalently, we can see the learned linear map.
>>> # The distance coincides with the euclidean distance after applying the linear map.
>>> L = nca.transformer()
>>> L
array([[ 0.77961901, -0.01911998, -0.35862791, -0.23992081],
       [-0.04424249,  1.80747788, -0.29365598, -0.25821444],
       [-0.69744415, -0.57288453,  2.16895076,  1.35252555],
       [-0.46668713, -0.48755353,  1.25732916,  2.20913531]]))

>>> # Finally, we can obtain the transformed data ...
>>> Lx = nca.transform()
>>> Lx[5,:]
array([[ 3.39595632,  2.9388401, -1.88738485, -1.85383882],
       [ 3.88626621,  2.33384985, -1.17887375, -1.1789864],
       [ 9.886781,   2.57431198, -1.06855681, -1.06854883],
       [ 2.94100652,  2.41813313, -1.05833389, -1.30275593],
       [ 3.27815332,  2.93403684, -1.88384889, -1.85654046]])
```

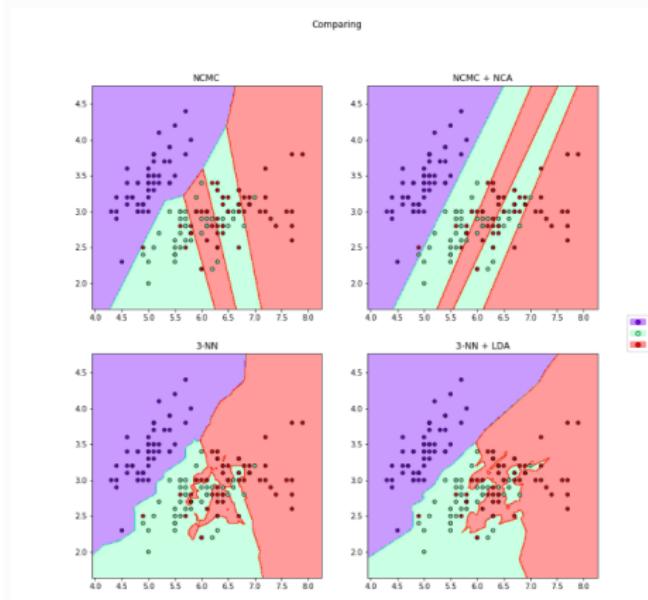
>>> # ... or transform new data.

```
>>> X_ = np.array([[1,0,0,0,0,0],[1,0,1,0,0,0],[1,0,1,0,1,0,0,0]])
>>> Lx_ = nca.transform(X_)
>>> Lx_
array([[ 0.77961901, -0.04424249, -0.69744415, -0.46668713],
       [ 0.76649063,  0.9630484, -1.18032868, -0.94824066],
       [ 0.49186212,  0.66368281,  0.98062208,  0.30968885 ]])
```

Funcionalidades adicionales

Dibujando regiones de clasificadores basados en distancias

```
>>> # We can compare different algorithms or distances together in the same figure
>>> f7 = dml_multiplot(X[:,[0,1]],y,nrow=2,ncol=2,ks=[None,None,3,3],
>>>           clfs=[ncmc,ncmc,None,None],kmis=[None,nca,None,lda],
>>>           transforms=[False,False,False,False],title="Comparing",
>>>           subtitles=["NCMC","NCMC + NCA","3-NN","3-NN + LDA"],
>>>           cmap="rainbow",figsize=(12,12))
```



Funcionalidades adicionales

Clasificadores basados en distancias

(Ampliación de los de Scikit-Learn)

```
>>> # The NCNC Classifier works like every ClassifierMixin.
>>> ncnc = NCNC_Classifier([centroids_num=2])
>>> ncnc.fit(X,y)
>>> ncnc.score(X,y)
0.9533333333333333

>>> # To learn a distance to use with NCNC Classifier, and with any other distance classifier
>>> # we can use pipelines.
>>> from sklearn.pipeline import Pipeline
>>> dml_ncnc = Pipeline([('nca',nca),('ncmc',ncmc)])
>>> dml_ncnc.fit(X,y)
>>> dml_ncnc.score(X,y)
0.9700000000000001
```

Estimación de hiperparámetros

```
>>> # Using cross validation we can tune parameters for the DML algorithms.
>>> # Here, we tune the NCA algorithm, with a fixed parameter learning_rate='constant'.
>>> # The parameters we tune are num_dims and eta0.
>>> # The metrics we use are 3-NN and 5-NN scores, and the final expectancy metadata of NCA.
>>> # A 5-fold cross validation is done twice to obtain the results.
>>> best, nca_best, detailed = tune_nca(num_dims=3, learning_rate='constant',
>>>                                tune_args={'num_dim': [3, 4], 'eta0': [0.001, 0.01, 0.1]},
>>>                                metrics=[3, 5, 'final_expectance'],
>>>                                n_folds=5, n_reps=2, seed=28, verbose=True)
>>> # Now we can compare the results obtained for each case.
>>> results
```

	3-NN	5-NN	final_expectance
{'num_dim': 3, 'eta0': 0.001}	0.963333	0.970690	0.891095
{'num_dim': 3, 'eta0': 0.01}	0.966667	0.963333	0.916248
{'num_dim': 3, 'eta0': 0.1}	0.956667	0.903333	0.925043
{'num_dim': 4, 'eta0': 0.001}	0.956667	0.933333	0.927239
{'num_dim': 4, 'eta0': 0.01}	0.956667	0.963333	0.922415
{'num_dim': 4, 'eta0': 0.1}	0.950099	0.963333	0.947319

rDML

- *Distance Metric Learning algorithms for R.*
- Wrapper en R para pyDML.
- Pone los algoritmos y funcionalidades adicionales de pyDML a disposición del lenguaje R.
- Herramienta de interacción: biblioteca `reticulate`.



Ejemplo

Aprendiendo distancias con rDML

First, we load the needed libraries.

```
library(datasets)
library(rDML)
```

We will use the iris dataset.

```
# Loading dataset
data(iris)
X = iris[,1:4]
y = iris[,5],1]
```

We construct the NCA transformer, and we fit it with the data in Iris.

```
# DML construction
nca = NCA()

# Fitting algorithm
nca$fit(X,y)
```

Once fitted, we can look at the algorithm metadata.

```
# We can look at the algorithm metadata after fitting it
meta = nca$metadata()
meta

## #num_iters
## [1] 2
##
## $initial_expectation
## [1] 0.8307717
##
## $final_expectance
## [1] 0.9543913
```

Also we can see the metric or the transformed we have learned.

```
# We can see the metric the algorithm has learned
M = nca$metric()
M

## ## [1,] 1.248526 0.448331 -1.818768 -1.664831
## [2,] 0.448331 1.248526 0.448331 -1.818768
## [3,] -1.818768 -1.664831 3.478283 4.644449
## [4,] -1.664831 -1.621492 4.644449 5.510808
```

#Equivalent, we can see the learned linear map
L = nca\$transformer()
L

```
## ## [1,] 1.11      1.82      1.93      1.41
## [2,] 0.895494 -0.412286 -0.270586 -0.123228
## [3,] 0.04871589 1.63398890 -0.3568736 -0.3212280
## [4,] -0.24982909 -0.44370160 2.0097958 1.1765839
## [5,] -0.41650226 -0.37170808 1.0333693 1.9964158
```

We can use the transformer to map data to the space defined by the learned distance.

```
# Finally, we can obtain the transformed data ...
Lx = nca$transform()
Lx[1:5,]
```

```
## ## [1,] 1.11      1.82      1.93      1.41
## [2,] 0.895494 3.373981 -1.905546 -1.379140
## [3,] 3.397400 2.837074 -0.965380 -1.309985
## [4,] 3.744631 3.073798 -1.1526161 -1.404363
## [5,] 3.401480 2.892182 -0.6416239 -1.118668
## [6,] 3.978998 3.474339 -1.2948878 -1.574660
```

```
# ... or transform new data.
X_ = matrix(nrow = 3, ncol = 4, data = c(1,0,0,0,
                                         1,1,0,0,
                                         1,1,1,0))
Lx_ = nca$transform(X_)

## ## [1,] 1.11      1.82      1.93      1.41
## [2,] 0.895494 -0.412286 -0.270586 -0.123228
## [3,] 0.04871589 1.63398890 -0.3568736 -0.3212280
## [4,] -0.24982909 -0.44370160 2.0097958 1.1765839
## [5,] -0.41650226 -0.37170808 1.0333693 1.9964158
```

Experimentación

Descripción

- Conjunto de experimentos para evaluar los algoritmos de aprendizaje de métricas de distancia.
- Experimentos realizados con la biblioteca pyDML.
- Evaluación: resultados de la clasificación con diferentes clasificadores basados en distancias.
- 34 datasets.
- 10-Fold Cross Validation y normalización previa.

Experimentos

- ① Algoritmos a dimensión máxima con k-NN.
- ② Algoritmos basados en centroides con sus correspondientes clasificadores.
- ③ Algoritmos basados en kernels con k-NN.
- ④ Algoritmos de reducción de dimensionalidad con k-NN.

Algunos resultados

	Euclidean	LDA	ITML	DMLMJ	NCA	LMNN	LSI	DML_eig	MCML	LDML
appendicitis	0.833939	0.852273	0.860455	0.825606	0.850455	0.842273	0.863030	0.862273	0.851364	0.842273
balance	0.808237	0.899236	0.894398	0.819113	0.958415	0.817523	0.928073	0.894502	0.873760	0.889554
bupa	0.654622	0.646555	0.628151	0.677647	0.599412	0.634286	0.628403	0.612017	0.574286	0.585462
cleveland	0.546833	0.550267	0.552362	0.563696	0.543678	0.580349	0.572220	0.582941	0.578577	0.597284
glass	0.701514	0.623153	0.654929	0.704158	0.691767	0.706733	0.623567	0.626371	0.585010	0.606334
hepatitis	0.832540	0.860913	0.881548	0.889484	0.832540	0.841865	0.913095	0.917659	0.882937	0.854762
ionosphere	0.855000	0.839472	0.886204	0.860551	0.908431	0.885962	0.876807	0.874118	0.863007	0.851232
iris	0.953333	0.953333	0.973333	0.966667	0.966667	0.940000	0.980000	0.960000	0.946667	0.960000
monk-2	0.965537	0.956129	0.935239	0.972460	1.000000	0.981657	1.000000	0.990909	0.967696	0.949577
newthyroid	0.953896	0.958658	0.939827	0.944805	0.972294	0.972511	0.963420	0.962987	0.958225	0.967532
sonar	0.837013	0.778225	0.812056	0.836147	0.870390	0.874242	0.850671	0.797554	0.856342	0.788680
wine	0.960675	0.988889	0.977376	0.966230	0.988235	0.983299	0.966230	0.976722	0.983299	0.988889
movement_libras	0.813944	0.664214	0.799221	0.864980	0.831939	0.802010	0.744055	0.787242	0.807313	0.736096
pima	0.739662	0.752597	0.714969	0.742259	0.737013	0.727837	0.739576	0.726658	0.723975	0.724009
vehicle	0.712582	0.762329	0.751621	0.755180	0.755008	0.675792	0.666690	0.650105	0.736938	0.717077
vowel	0.978788	0.977778	0.953535	0.980808	0.980808	0.977778	0.947475	0.675758	0.873737	0.909091
wdbc	0.971679	0.966446	0.966415	0.964815	0.970079	0.963028	0.968292	0.950717	0.964876	0.943821
wisconsin	0.967859	0.967794	0.959035	0.967859	0.964831	0.966324	0.972206	0.970715	0.954623	0.966346
banana	0.855585	0.646943	0.855603	0.856520	0.858389	0.858300	0.851794	0.687859	0.610209	0.631972
digits	0.986666	0.968319	0.972841	0.983400	0.989427	0.980696	0.910254	0.816843	0.968848	0.981638
letter	0.720827	0.796751	0.719501	0.820461	0.861029	0.716247	0.549663	0.321460	0.753477	0.637251
magic	0.805086	0.736179	0.806144	0.807189	0.814527	0.794554	0.792421	0.752540	0.776673	0.695187
optdigits	0.977732	0.951272	0.966963	0.976129	0.975930	0.984048	0.930608	0.802271	0.959112	0.959164
page-blocks	0.949555	0.967908	0.961477	0.950490	0.957739	0.943965	-	0.952300	0.964205	0.940490
phoneme	0.799285	0.724324	0.777087	0.800228	0.793686	0.794612	0.766867	0.748339	0.763283	0.711248
ring	0.643274	0.710104	0.735260	0.645306	0.845951	0.661573	0.816211	0.722303	0.822361	0.564865
satimage	0.856497	0.838704	0.834163	0.864328	0.851141	0.855846	0.846520	0.813018	0.817149	0.550118
segment	0.902041	0.937075	0.926531	0.908163	0.918707	0.892857	0.885374	0.906803	0.931973	0.871088
spambase	0.865481	0.887127	0.876682	0.852574	0.915487	0.907075	0.911141	0.904665	0.904758	0.898278
texture	0.961818	0.998182	0.975455	0.985455	0.980000	0.921818	0.940000	0.900909	0.974545	0.871818
thyroid	0.931948	0.945086	0.940297	0.936125	0.939569	0.931958	0.935421	0.948568	0.932006	0.958383
titanic	0.758392	0.780464	0.760933	0.758734	0.676411	0.696411	-	0.733135	0.725301	0.734112
twonorm	0.959500	0.975018	0.966915	0.956122	0.979072	0.975698	0.977045	0.981099	0.973000	0.980424
winequality-red	0.586563	0.573370	0.582888	0.586076	0.576614	0.577234	0.580989	0.529240	0.561132	0.547197
AVG RANKING	5.661765	5.426471	5.382353	4.544118	3.661765	5.279412	5.382353	6.220588	6.279412	7.161765
AVG SCORE	0.842585	0.836326	0.847042	0.852640	0.863401	0.843287	0.840566	0.804135	0.835902	0.806213

Análisis de resultados

- ① Mejores resultados en clasificación k-NN: **NCA**, **LMNN** y **DMLMJ**.
- ② Buenos resultados de **NCMML** y **NCMC** sobre los clasificadores basados en centroides.
- ③ Algoritmos basados en kernels: gran variedad de posibilidades.
- ④ Ventajas de la reducción de dimensionalidad.

Introducción
oooooo

Matemáticas
oooooooooooo

Informática teórica
ooooooo

Informática práctica
oooooooooooo

Conclusiones y vías futuras
oo

1 Introducción

2 Matemáticas

3 Informática teórica

4 Informática práctica

5 Conclusiones y vías futuras

- Conclusiones y vías futuras

Conclusiones

Hemos estudiado el concepto de aprendizaje de métricas de distancia, con sus fundamentos teóricos y algunos de sus algoritmos. También se han recogido los algoritmos en un software, con el que se han elaborado distintos experimentos.

Vías futuras

- ① Incorporación de nuevos algoritmos.**
- ② Nuevos enfoques para el concepto de distancia.**
- ③ Kernelización de más algoritmos.**
- ④ Otros mecanismos de optimización.**

¡Gracias por su atención!

Juan Luis Suárez Díaz
jlsuarezdiaz@correo.ugr.es