INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Universidad de Oviedo

Introducción al Machine Learning

Ignacio Díaz Blanco

¡un problema *Big Data*!

- Cantidades ingentes de datos
  - Infinidad de sensores
  - Ubicuidad de la información
  - Heterogeneidad de la información
- Sistemas complejos
  - Procesos dinámicos
  - Conexión, interacción y acoplamiento
  - Muchas variables, problemas multivía
  - Interacción humana
  - Interacción con el tejido productivo

Problema:

Aprendizaje de modelos a partir de datos

Buscar estructuras en los datos y modelarlas

Referencias:
Big data: The next frontier for innovation, competition, and productivity.
http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation.

¿qué es el Aprendizaje Automático?

"Machine Learning"

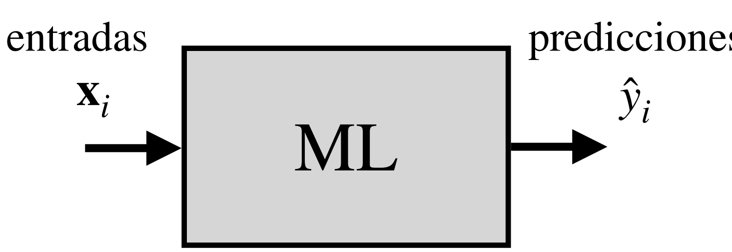
Universidad de Oviedo

Aprendizaje (definición formal)
Un programa de computador se dice que "aprende" de la experiencia E con respecto a una serie de tareas T y un índice de rendimiento P, si dicho rendimiento P para las tareas T mejora con la experiencia E.

Tom Mitchell

$T \rightarrow$ ej diagnóstico fallos (clasificación), estimación de variables como la temperatura en un motor (regresión), detección de anomalías, predicción de posición de objetos móviles en robótica etc.

$E \rightarrow$ datos de entrenamiento $\{\cdots, (\mathbf{x}_i, y_i), \cdots\}$



$P \rightarrow$ función de coste, ej. suma de errores cuadráticos

$$\sum_i (y_i - \underbrace{f(\mathbf{x}_i, \mathbf{W})}_{\hat{y}_i})^2$$

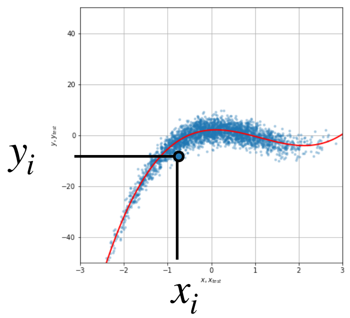
Paradigmas de aprendizaje

Aprendizaje Supervisado

utiliza datos etiquetados

con entradas $\mathbf{x}_i$ y sus correspondientes respuestas o *etiquetas* y_i

$$\{\cdots, (\mathbf{x}_i, y_i), \cdots\}$$



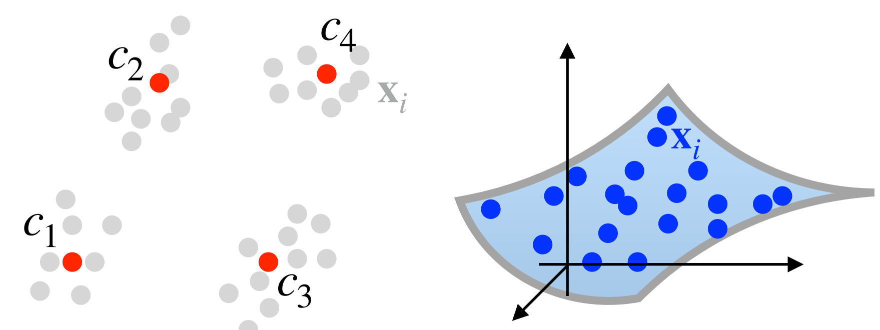
Aprendizaje No Supervisado

utiliza datos sin etiquetar, solo las entradas:

$$\{\cdots, \mathbf{x}_i, \cdots\}$$

debe ser capaz de detectar patrones "interesantes"

sin darle guías… solo con a partir de los datos



Aprendizaje por Refuerzo

Un sistema de aprendizaje (agente) interactúa con el mundo

y aprende a maximizar una señal de "recompensa"

Tipos de datos de entrada

n = muestra, k = timestep, descriptor, i, j = coordenadas en imagen, c = canal

Escalares
(n)

x

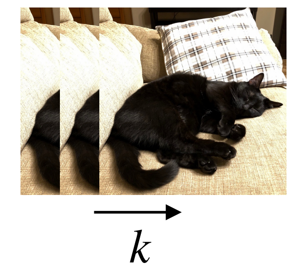
vectores
(n, c)

$(\cdots, x_c, \cdots)$

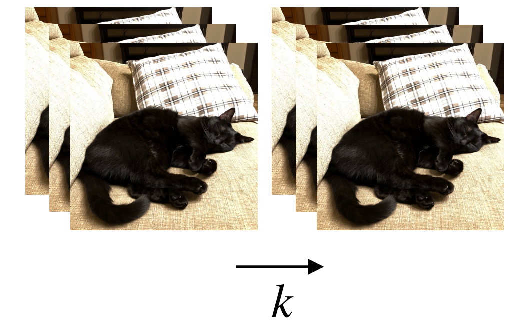
imágenes
(n, i, j)

imágenes multicanal
(n, i, j, c)

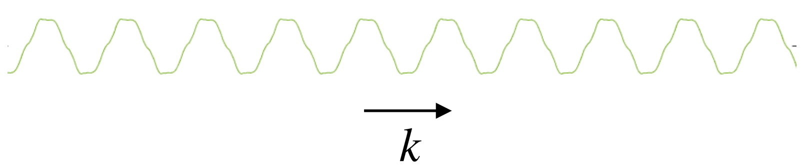
vídeos
(n, i, j, k)



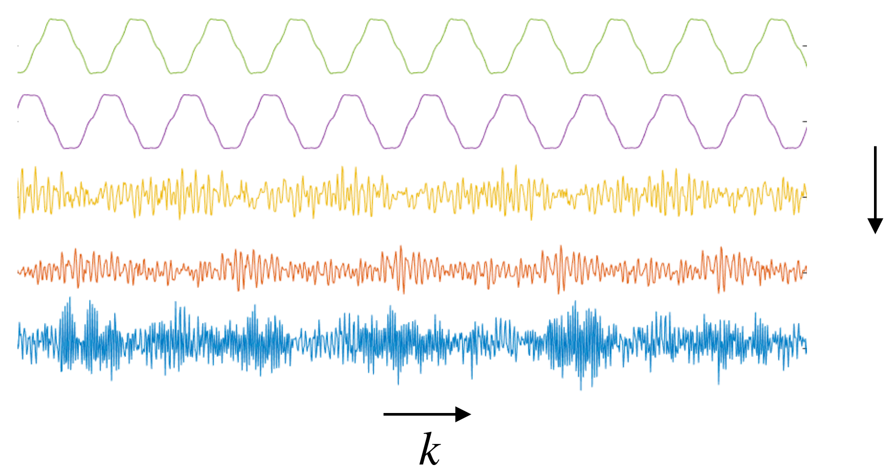
vídeos multicanal
(n, i, j, k, c)



series temporales
(n, k)



series temporales (multicanal)
(n, k, c)



Flujo de trabajo en ML

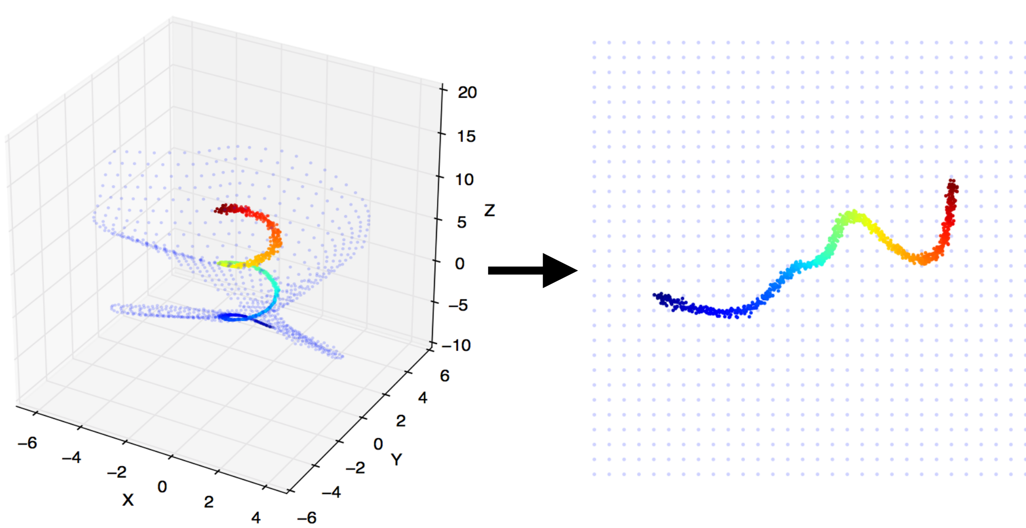
1. Planteamiento
    - ¿qué tareas T debo resolver?
    - ¿qué datos E uso?
    - adquirir conocimiento del problema
    - visualización de datos, análisis exploratorio
2. Organizar los datos
    - gestión de datos faltantes (eliminación, imputación)
    - reducción de muestras:
        - filtrado
        - agregación
    - reducción de atributos:
        - reducción de la dimensionalidad
        - selección de características relevantes
        - extracción de características (descriptores)
3. Elegir modelo, función de coste, regularización
4. Optimización
5. Búsqueda de hiperparámetros

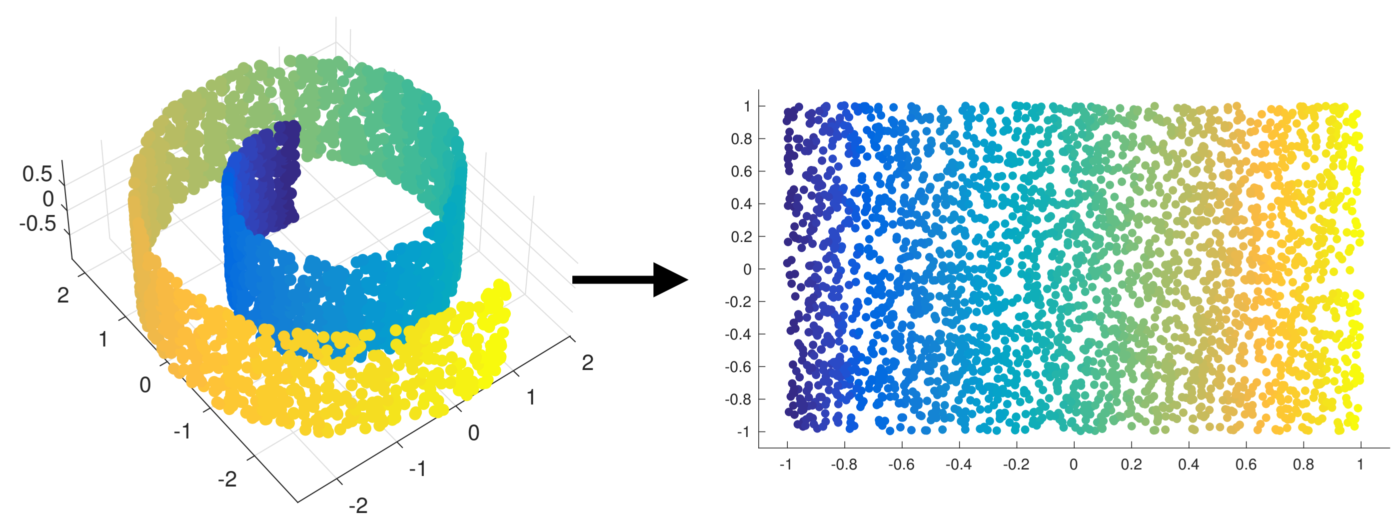
Machine learning

detectar estructura en los datos
desenredo de información (“disentangling”)

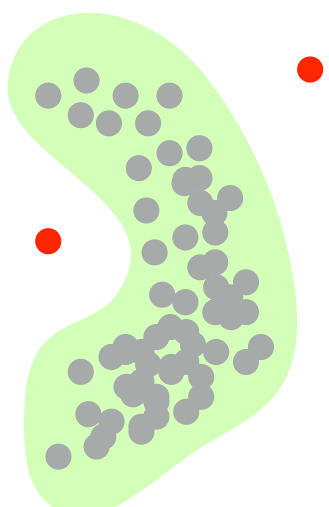
Universidad de Oviedo

reducción de la dimensionalidad

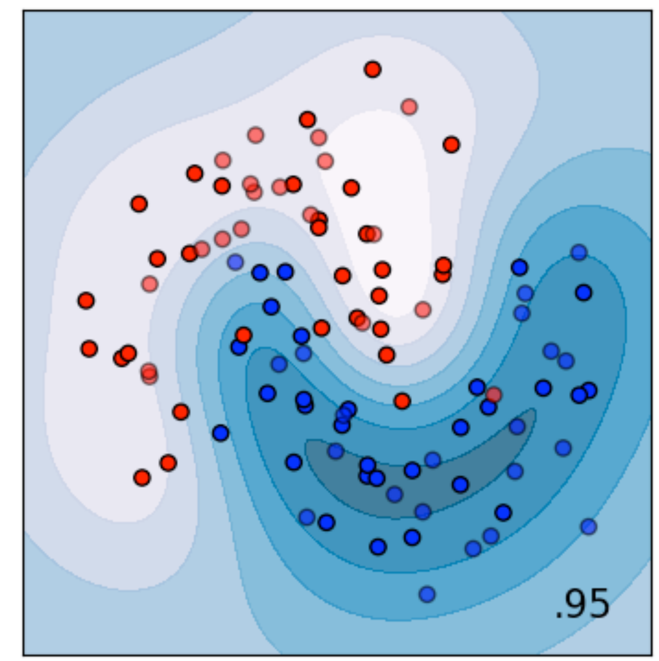




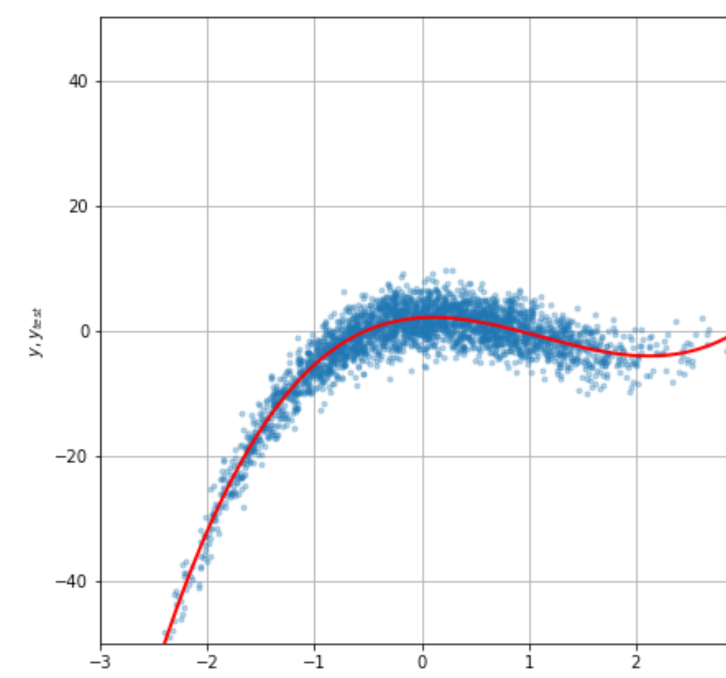
detección de anomalías



clasificación



regresión



tareas de ML

Regresión

Clasificación

Reducción de la dimensionalidad

Regresión

Regresión

Modelo y entrenamiento

$$\mathbf{y} = F(\mathbf{x}, \mathbf{W})$$

Conjunto de datos de entrenamiento

$$\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$$

objetivo del entrenamiento

W t.q. $\underbrace{\sum_i \|\mathbf{y}_i - F(\mathbf{x}_i, \mathbf{W})\|^2}_{E(W)}$ sea mínimo

clasificación: $\mathbf{y}_i = \{(0.3,1),0.5,1),(-0.2,0),\ldots,(-0.3,0)\}$

regresión: $\mathbf{y}_i = \{(0.2,1.1),1.5,-1.2),(-0.23,0.12),\ldots,(-0.12,0.43)\}$

$$\downarrow$$

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} E(\mathbf{W})$$

Regresión

Descenso del gradiente

Si definimos el error de un modelo como

$$E = \sum_i \|\mathbf{y}_i - F(\mathbf{x}_i, \mathbf{W})\|^2$$

dado que $\mathbf{x}_i, \mathbf{y}_i$ son conocidos y fijos (son los ejemplos) tenemos que

$$E = E(\mathbf{W})$$

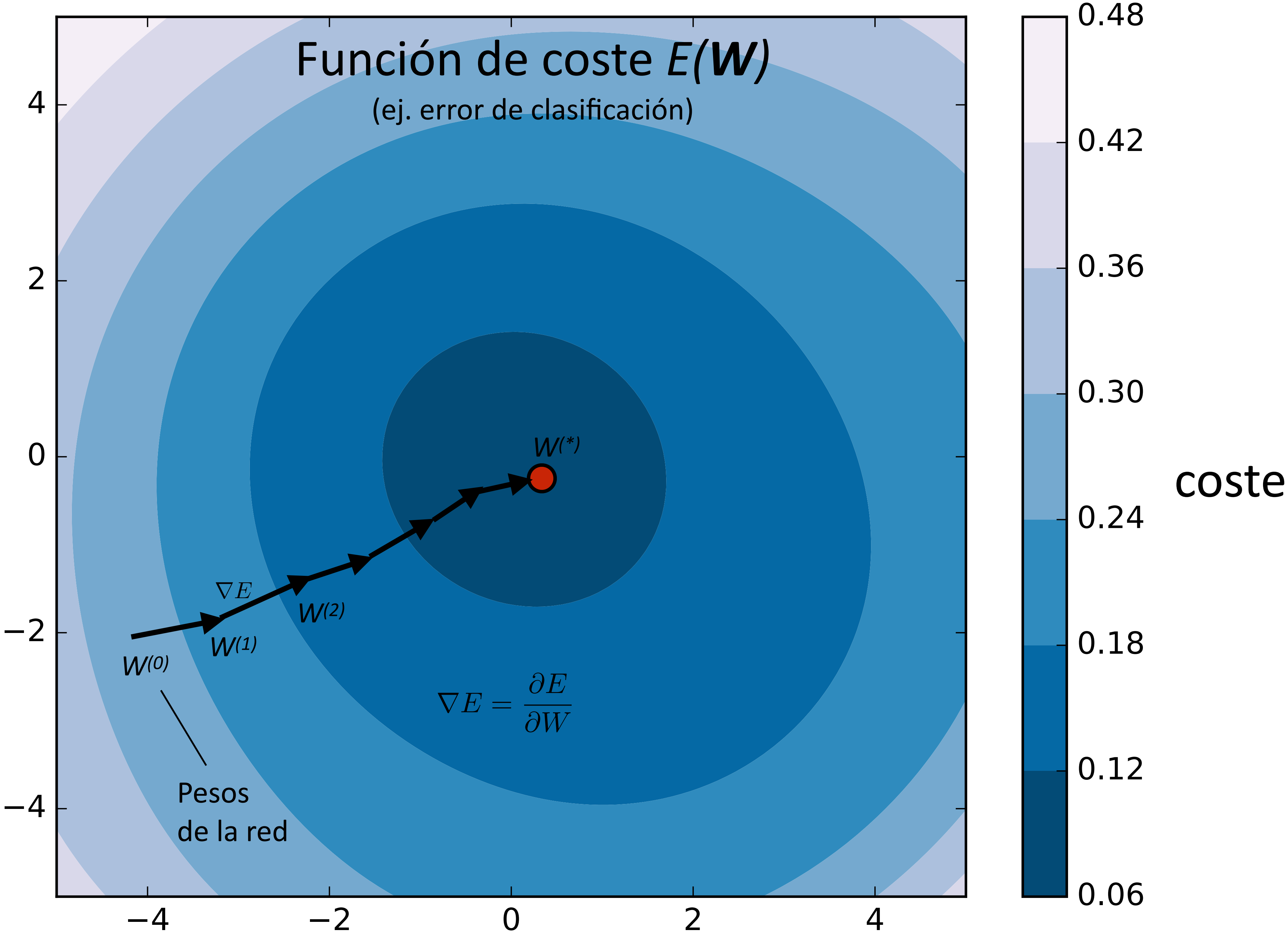
¡El gradiente del error

$$\nabla E = \frac{\partial E}{\partial \mathbf{W}}$$

nos dice en qué dirección debemos mover los pesos para disminuir el error!

$$\mathbf{W}^{(k)} \leftarrow \mathbf{W}^{(k-1)} - \mu \frac{\partial E}{\partial \mathbf{W}}$$

Regresión
Descenso del gradiente



Regresión

Optimización directa

Modelo lineal genérico (multivariable)

$$\mathbf{Y} = \mathbf{X}\mathbf{W}$$

datos de salida (targets)

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} \in \mathbb{R}^{nq}$$

pesos (coeficientes del modelo)

$$\mathbf{W} \in \mathbb{R}^{pq}$$

datos de entrada

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{np}$$

Ridge Regression

$$\hat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X} - \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|^2 + \lambda\|\mathbf{W}\|^2$$

↑
regularización
de Tikhonov

ejemplo en Python ↓

```
from sklearn.linear_model import Ridge
import numpy as np
n_samples, n_features = 10, 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
clf = Ridge(alpha=1.0)
clf.fit(X, y)
```

Regresión

Optimización directa (regresión dispersa)

Lasso

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|^2 + \alpha\|\mathbf{W}\|_1$$

```
from sklearn import linear_model
clf = linear_model.Lasso(alpha=0.1)
clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
print(clf.coef_)
print(clf.intercept_)
```

← ejemplo en Python

Elastic Net

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\mathbf{XW} - \mathbf{Y}\|^2 + \lambda_1\|\mathbf{W}\|_1 + \lambda_2\|\mathbf{W}\|^2$$

```
X, y = make_regression(n_features=2, random_state=0)
regr = ElasticNet(random_state=0)
regr.fit(X, y)
print(regr.coef_)
print(regr.intercept_)
print(regr.predict([[0, 0]]))
```

← ejemplo en Python

Regresión
modelos multivariable

Modelo multivariable
y depende de varios factores $x_1, \ldots, x_p$ (regresores)

$$y = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n + \beta$$

$$\mathbf{Y} = \mathbf{XW}$$

$$\mathbf{Y} = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} x_1^1 & x_2^1 & \ldots & x_p^1 & 1 \\ x_1^2 & x_2^2 & \ldots & x_p^2 & 1 \\ \vdots & & & & \\ x_1^n & x_2^n & \ldots & x_p^n & 1 \end{pmatrix} \qquad \mathbf{W} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \\ \beta \end{pmatrix}$$

salidas

p regresores

↑
término independiente (bias)

$p + 1$ coeficientes

Regresión

modelos no lineales

Modelo no lineal (lineal en los parámetros)

y como combinación lineal de funciones no lineales de $x_1, \dots, x_p$

$$y = \alpha_1 f_1(\mathbf{x}) + \alpha_2 f_2(\mathbf{x}) + \dots + \alpha_n f_n(\mathbf{x}) + \beta$$

regresores no lineales

$$f_i(\mathbf{x}) = f_i(x_1, x_2, \dots, x_p)$$

ejemplos de $f_i(\mathbf{x})$:

$x_3 x_5,\ \cos(x_3),\ \cos(5x_4),$

$\sqrt{(x_1 x_4)},\ x_1^2 + x_3^2,$

$x_1 x_2 x_3^2 x_4 x_5$

$$\mathbf{Y} = \mathbf{X}\mathbf{W}$$

$$\mathbf{Y} = \begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \dots & f_p(\mathbf{x}) & 1 \\ f_1(\mathbf{x}) & f_2(\mathbf{x}) & \dots & f_p(\mathbf{x}) & 1 \\ \vdots & & & & \\ f_1(\mathbf{x}) & f_2(\mathbf{x}) & \dots & f_p(\mathbf{x}) & 1 \end{pmatrix} \qquad \mathbf{W} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \\ \beta \end{pmatrix}$$

salidas

p regresores

↑ término independiente (bias)

$p + 1$ coeficientes

Modelos dinámicos lineales (discretos)

Función de transferencia (SISO discreto)

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \ldots b_m z^{-m}}{1 - a_1 z^{-1} - \ldots - a_n z^{-n}}$$

ecuación en diferencias

$$y(k) = a_1 y(k-1) + \ldots + a_n y(k-n) + b_0 u(k) + \ldots + b_m u(k-m)$$

modelo de regresión multivariable

→ se suponen disponibles datos para muchos valores de k

$$\begin{bmatrix} \vdots \\ y(k) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & \vdots & & & \\ y(k-1) & \cdots & y(k-n) & u(k) & \cdots & u(k-m) \\ & & \vdots & & & \end{bmatrix} \begin{bmatrix} -a_1 \\ \vdots \\ -a_n \\ b_0 \\ \vdots \\ b_m \end{bmatrix}$$

$$\mathbf{Y} = \mathbf{XW}$$

si se conocen, utilizar
las derivadas como regresores

sistema SISO continuo

$$\{x(t)\} \rightarrow \{y(t)\}$$

$$y^{(m)}(t) = \alpha_{m-1} y^{(m-1)}(t) + \ldots + \alpha_1 \dot{x}(t) + \beta_r u^{r)}(t) + \ldots + b_1 \dot{u}(t) + \beta_0 u(t) + \gamma$$

regresores con
las derivadas

Problema práctico
muy sensible al ruido por las derivadas

Identificación de Sistemas

Modelos dinámicos no lineales (NARX)

modelo NARX (*Non-linear Autoregresive eXogenous*)

$$y(k) = f(y(k-1), \ldots y(k-n_y), u(k), \ldots u(k-n_u))$$

en general, puede aproximarse con regresores
de uno o varios retardos
una o varias variables de salida / entrada

$$\cdots \quad + \quad \alpha_1 y(k-2)y(k-5) \quad + \quad \alpha_2 y(k-6)u(k) \quad + \quad \alpha_3 \cos(y(k)^2 u(k)) \quad + \ldots$$

INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

Universidad de Oviedo

Identificación de Sistemas

espacio de estados (SINDy)


Discovering governing equations from data by sparse identification of nonlinear dynamical systems

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, *113*(15), 3932-3937.

$$\frac{d}{dt}\mathbf{x} = f(\mathbf{x})$$

sistema
no lineal
genérico

$$f(\mathbf{x}) \approx \sum_k \theta_k(\mathbf{x})\xi_k = \Theta(\mathbf{x})\xi$$

aproximación
mediante
combinación lineal
de funciones no lineales

datos y targets
para m muestras
en formato matricial

$$\mathbf{X} = [\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots \mathbf{x}(t_m)]$$

$$\dot{\mathbf{X}} = [\dot{\mathbf{x}}(t_1), \dot{\mathbf{x}}(t_2), \ldots \dot{\mathbf{x}}(t_m)]$$

expansión no lineal
de los datos
en formato matricial

$$\Theta(\mathbf{X}) = [\mathbf{1}, \mathbf{X}, \mathbf{X}^2, \ldots \cos(\mathbf{X}) \ldots]$$

nos lleva a un
problema de regresión
lineal multivariable

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\boldsymbol{\Xi}$$

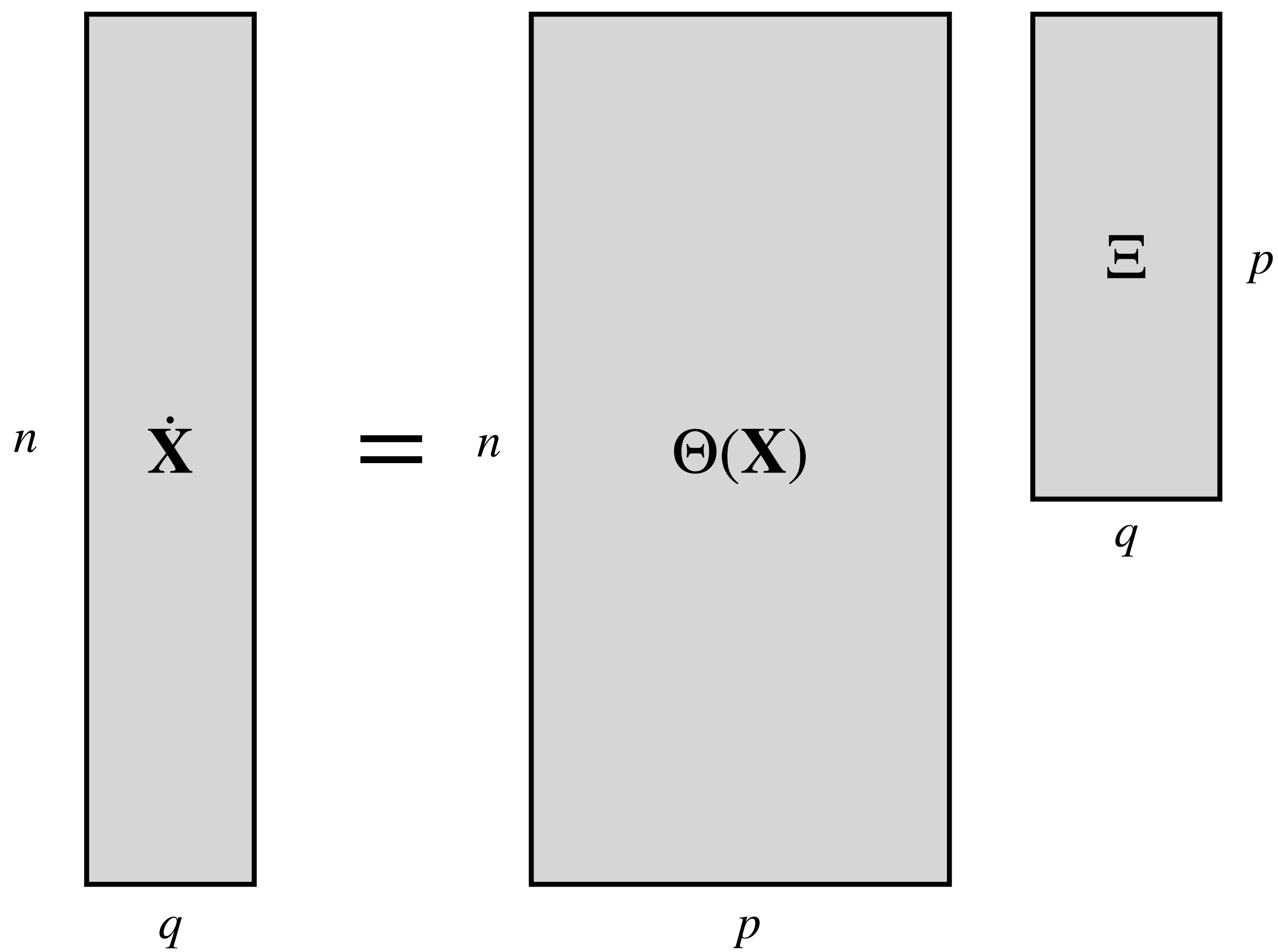
espacio de estados (SINDy)

(Brunton & Kutz, 2022)

$p =$
número de
funciones no lineales
de la expansión

$q =$
número de
variables
de estado

$n =$
muestras
disponibles



Identificación de Sistemas

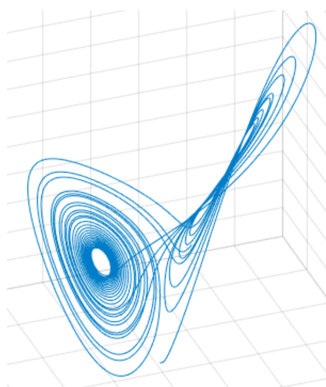
espacio de estados (SINDy)

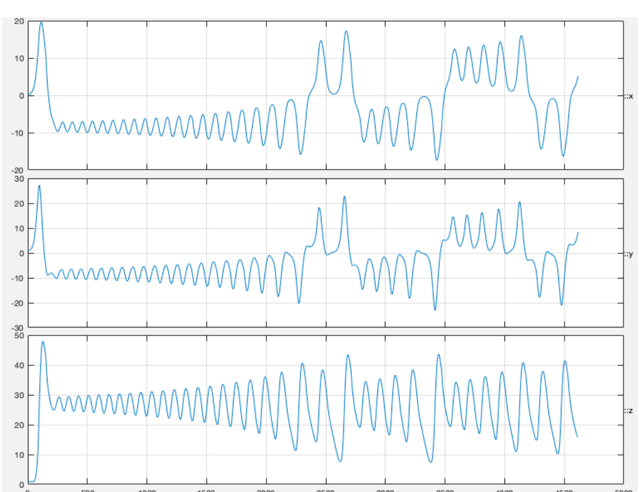
Ecuaciones de Lorenz

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$





$$\Theta(\mathbf{x}) = x, y, z, x^2, xy, xz, y^2, yz, z^2, x^3, x^2y, x^2z, xy^2, xyz, xz^2, y^3, y^2z, yz^2, z^3$$

$$\downarrow$$

$\dot{x}, \dot{y}, \dot{z},$ $x, y, \ \ldots xz, \ yz, \ \ldots \ , yz^2, z^3$ ξ_1, ξ_2, ξ_3

$$\underbrace{\dot{\mathbf{X}}}_{n \times q} = \underbrace{\Theta(\mathbf{X})}_{n \times p}\ \underbrace{\Xi}_{p \times q}$$

Identificación de Sistemas

espacio de estados (SINDy)

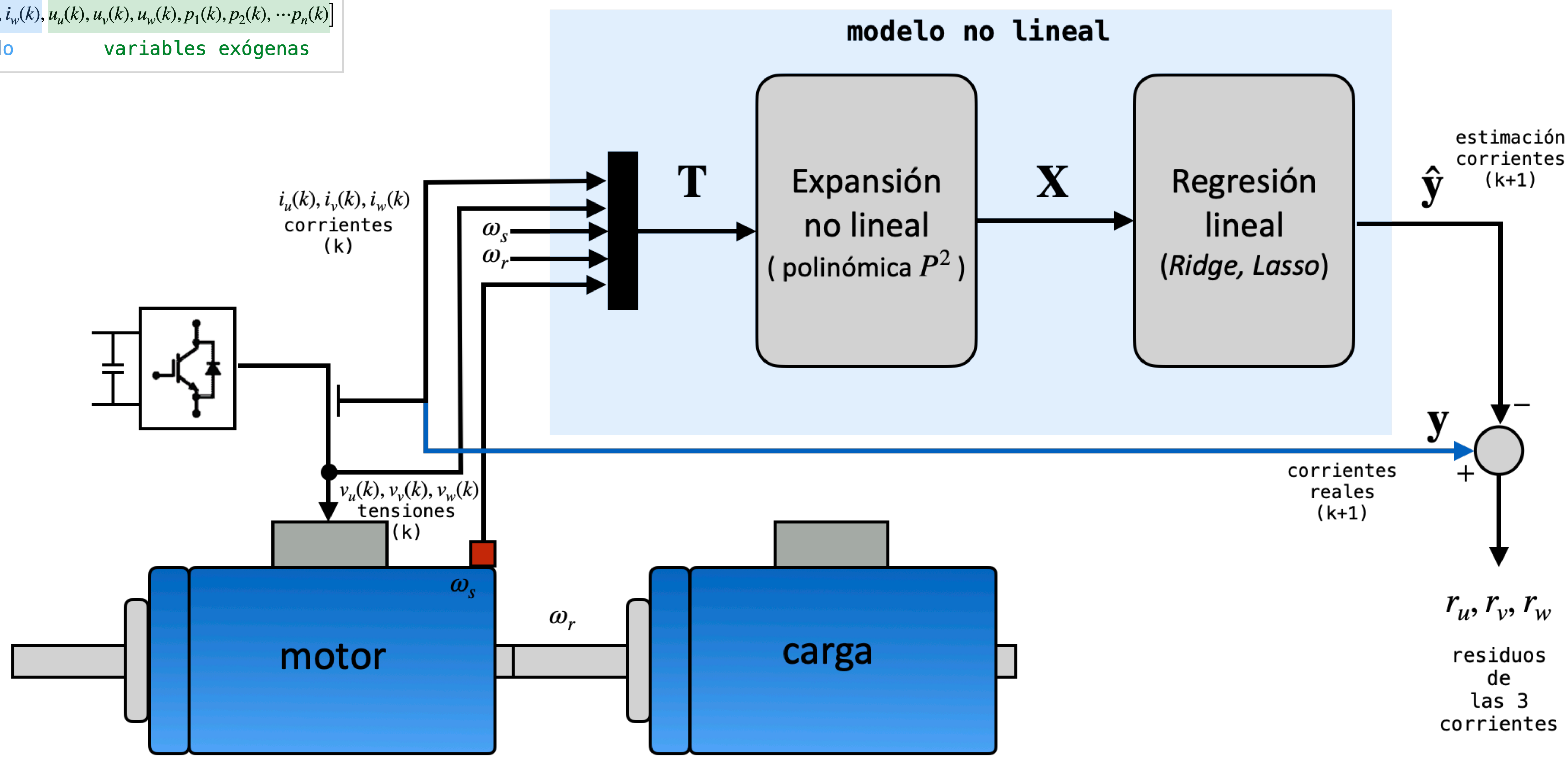
estado en (k+1)

$[i_u(k+1), i_v(k+1), i_w(k+1)]$

$$\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k))$$

$$\mathbf{T} = [\underbrace{i_u(k), i_v(k), i_w(k)}_{\text{estado}}, \underbrace{u_u(k), u_v(k), u_w(k), p_1(k), p_2(k), \cdots p_n(k)}_{\text{variables exógenas}}]$$

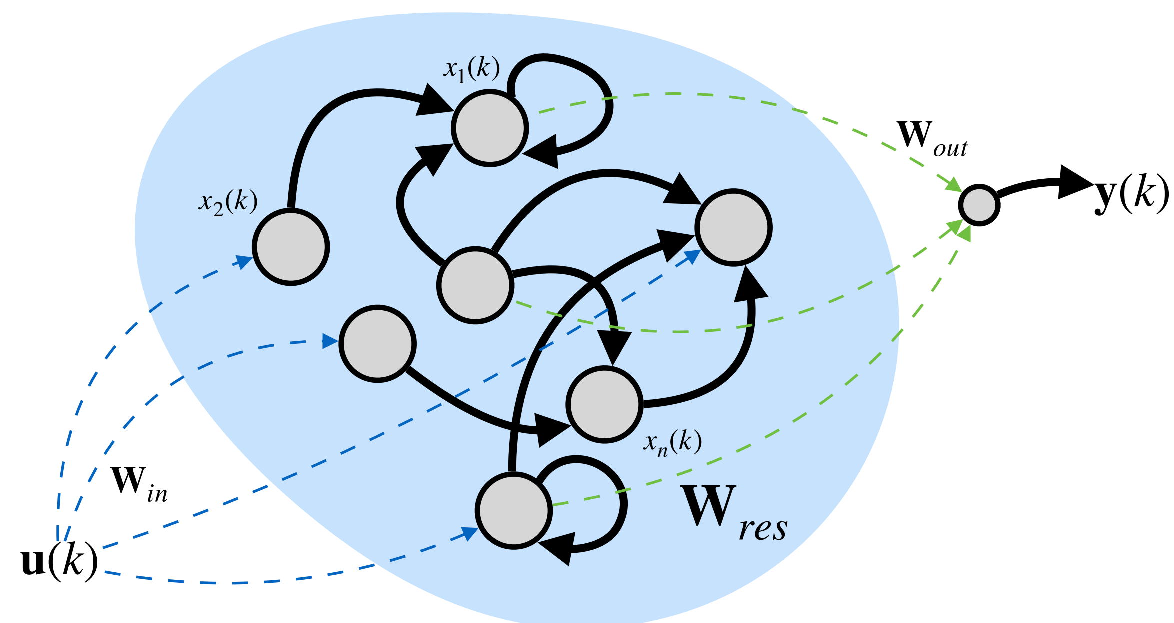
Ejemplo: detección de anomalías

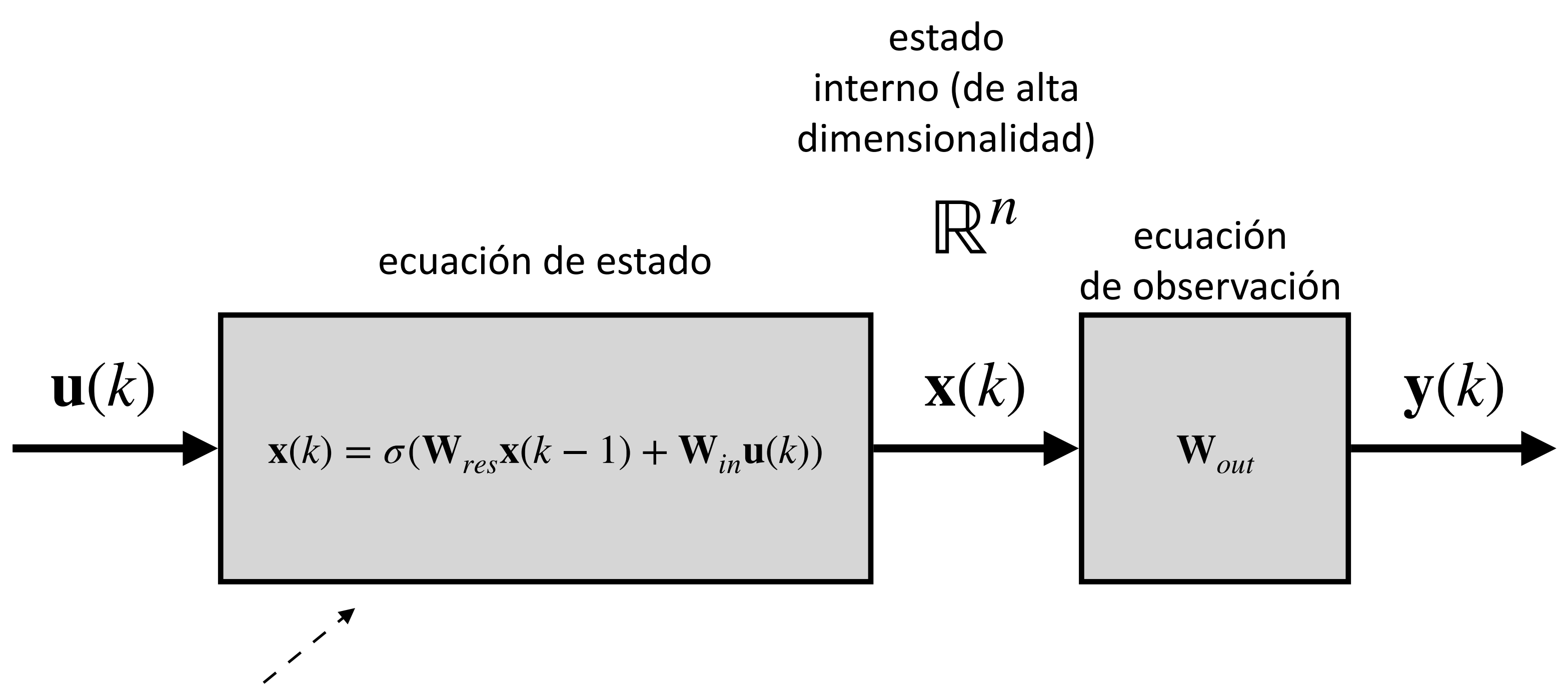


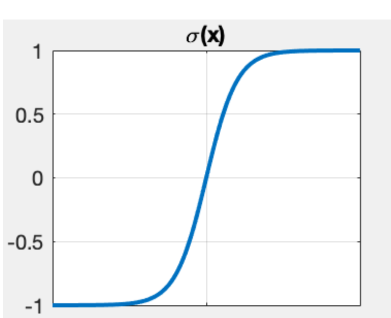
Identificación de Sistemas

Echo State Networks

$$\mathbf{x}(k) = \sigma(\mathbf{W}_{res}\mathbf{x}(k-1) + \mathbf{W}_{in}\mathbf{u}(k))$$
$$\mathbf{y}(k) = \mathbf{W}_{out}\mathbf{x}(k)$$





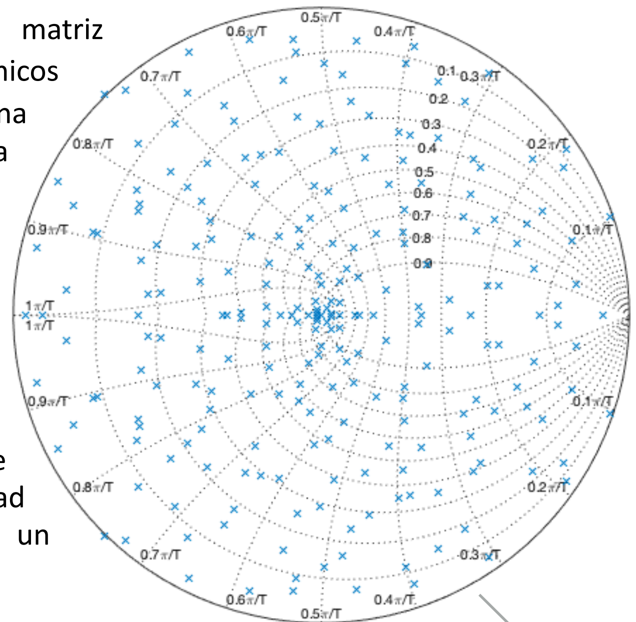


Identificación de Sistemas

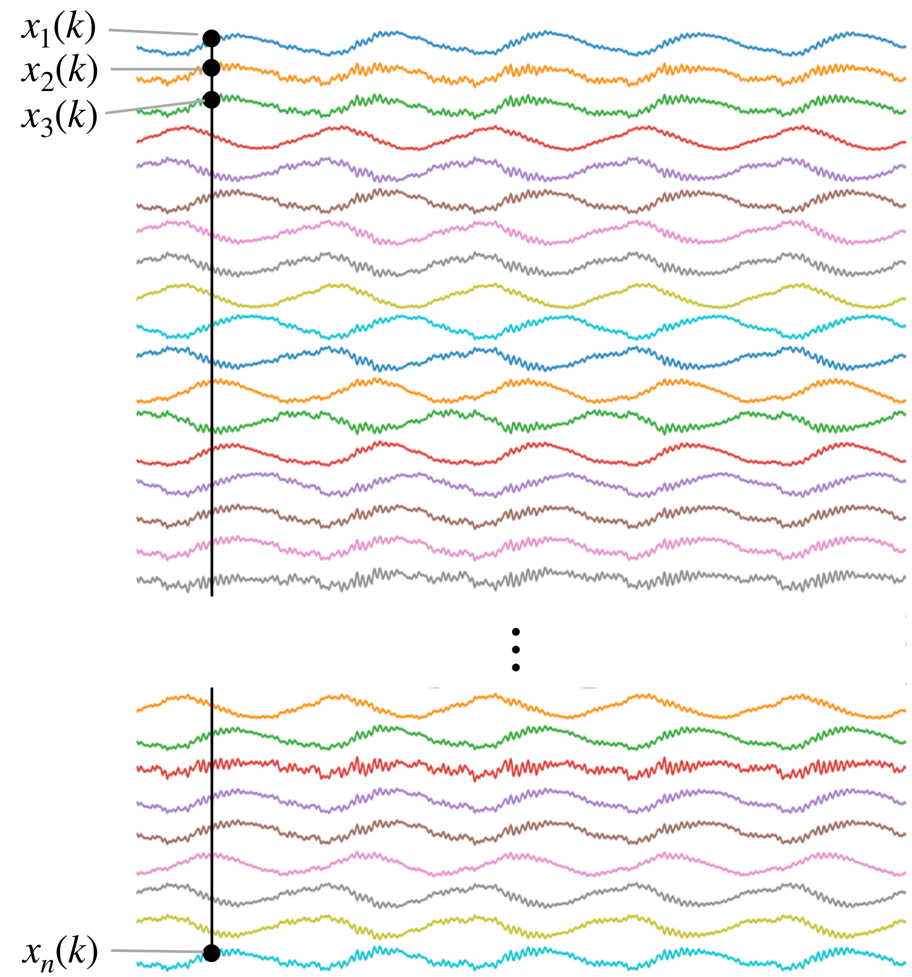
Echo State Networks

los valores propios de la matriz $\mathbf{W}_{res}$ son los modos dinámicos del sistema si $\sigma(\cdot)$ es una función lineal. Influyen en la naturaleza de los modos dinámicos no lineales cuando σ es sigmoidal.

Para generar un amplio espectro de dinámicas se hace $\mathbf{W}_{res}$ dispersa (mayoría de ceros) y se lleva al borde de estabilidad escalándola para alcanzar un radio espectral $\rho \approx 1$



matriz de reservorio, $\mathbf{X}$ (estados $\times$ muestras)



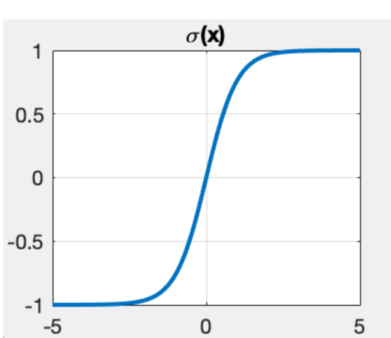
Reservorio
Contiene un número elevado n de modos dinámicos

$x_1(k), x_2(k), \ldots, x_n(k)$

generados no linealmente a partir de la entrada $\mathbf{u}(k)$

puede verse como una ***expansión no lineal*** de $\mathbf{u}(k)$ que la "disecciona" en muchas componentes no lineales

Entrada (excitación) $\mathbf{u}(k)$
señal exógena (puede ser multivariable)

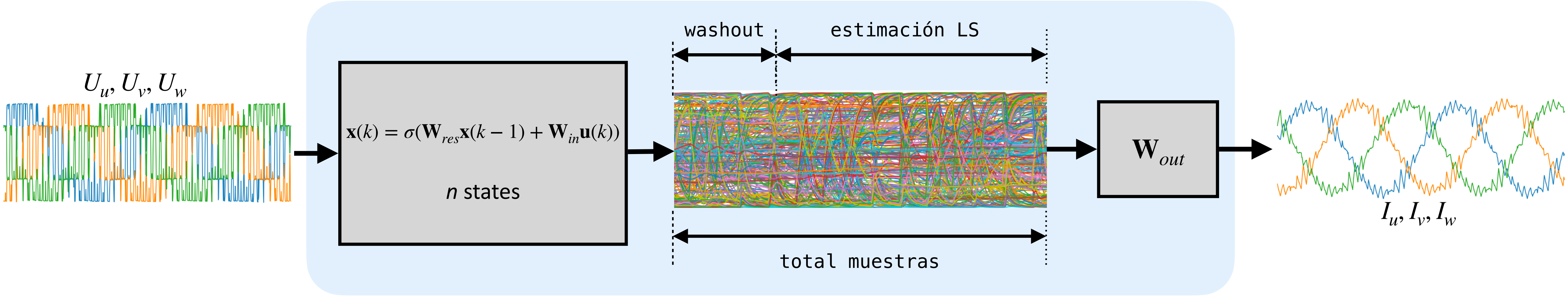


$\mathbf{u}(k)$ → $\mathbf{x}(k) = \sigma(\mathbf{W}_{res}\mathbf{x}(k-1) + \mathbf{W}_{in}\mathbf{u}(k))$ → $\mathbf{x}(k)$ → $\mathbf{W}_{out}$ → $\mathbf{y}(k)$

Salida
Cuando es dato (problema de identificación) permite estimar $\mathbf{W}_{out}$

$$\mathbf{W}_{out} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{Y}^T$$

Ejemplo: modelado tensión corriente



Dificultad de la generalización

de todos modos, ¡no suele ser tan fácil!

funciona bien en un punto de trabajo,
pero estos modelos suelen ser muy sensibles
si se cambia el punto de operación

usar modelos
más generales

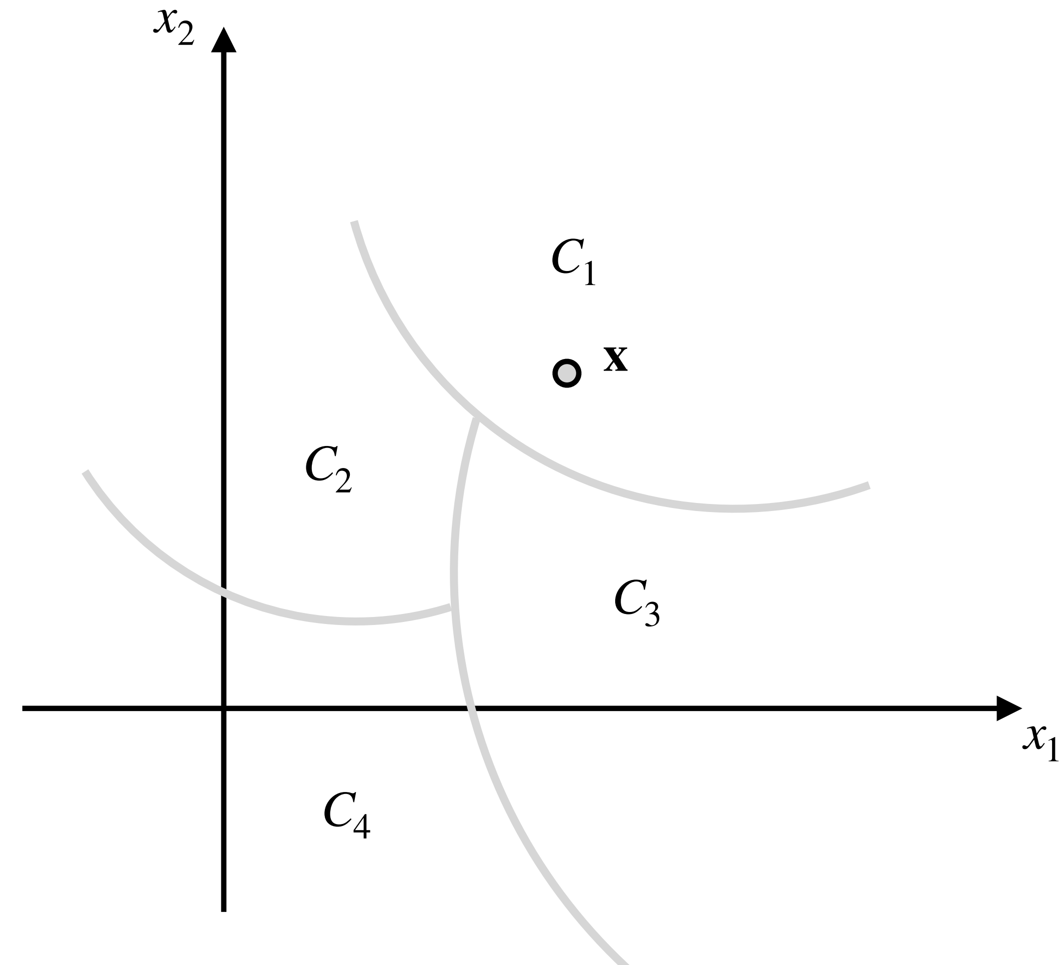
arquitecturas con modelos
en varios puntos de trabajo

Clasificación

El problema de clasificación

Objetivo:
asignar una nueva observación $\mathbf{x}$ a una clase C_i
de entre un conjunto de clases $\{C_1, C_2, \cdots, C_n\}$

En general, $\mathbf{x} = (x_1, x_2, \cdots x_p)$ suele ser un vector de características (*feature vector*) con descriptores x_i sensibles a las condiciones que producen el fallo



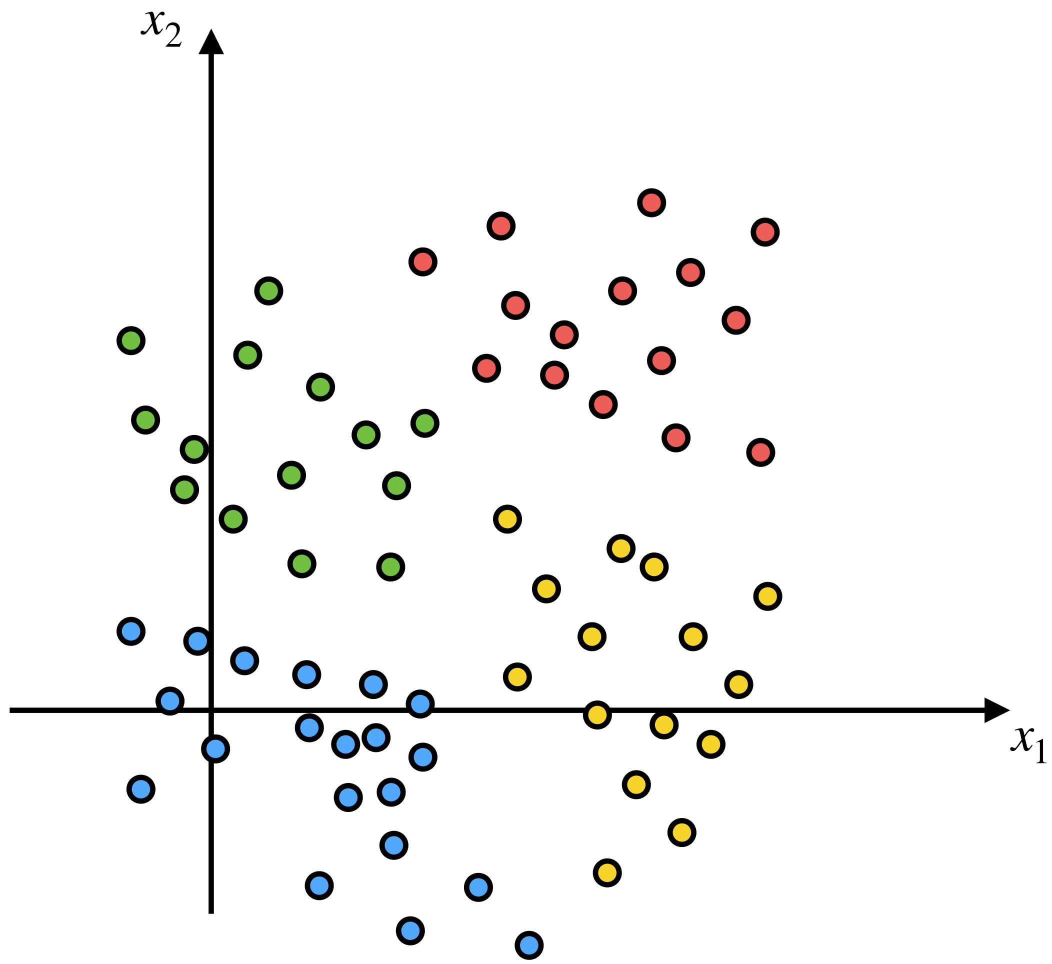
El problema de clasificación

Universidad de Oviedo

Se parte de datos etiquetados

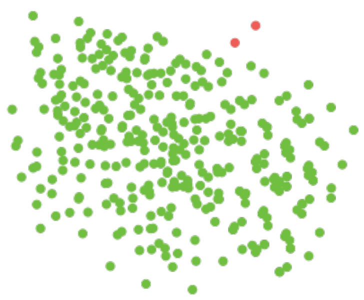
$$\{(\mathbf{x}_1, c_1), \ldots (\mathbf{x}_n, c_n)\}$$

$$c_i = \{\text{rojo, verde, amarillo, azul}\}$$



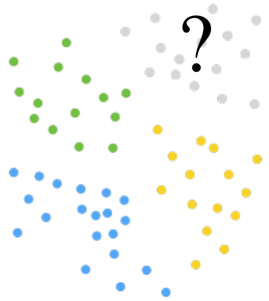
¡ requerir datos etiquetados es una **exigencia "fuerte"** !

A. Desbalanceo
Puede haber pocos datos etiquetados de una clase y muchos de otra



métodos
no supervisados

B. Falta de etiquetas
No siempre se dispone de datos para todos los fallos posibles



métodos
semisupervisados

Clasificador Naïve Bayes

teorema de Bayes

Posibilidades para estimar los componentes
(¡ no las únicas !)

$p(C_i) \approx \frac{\text{num datos de fallo}}{\text{num datos totales}}$ (o si los hay, estadísticos históricos de frecuencia del fallo)

$p(\mathbf{x} \mid C_i) \approx \text{normpdf}(\mathbf{x}, \mu_\text{i}, \sigma_\text{i})$
asumir distribución normal
calculando la media μ y la desviación típica σ
muestrales de los datos de la clase C_i

$p(\mathbf{x}) = \sum_{k=1}^{n} p(\mathbf{x} \mid C_k) P(C_k)$ (teorema de probabilidad total)

"prior"
probabilidad
de pertenecer a la clase C_i
sin otra consideración

densidad de probabilidad
condicional de $\mathbf{x}$
sabiendo que pertenece a C_i

"posterior"
probabilidad de pertenecer
a la clase C_i conocida $\mathbf{x}$

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i) P(C_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid C_i) P(C_i)}{\sum_{k=1}^{n} p(\mathbf{x} \mid C_k) P(C_k)}$$

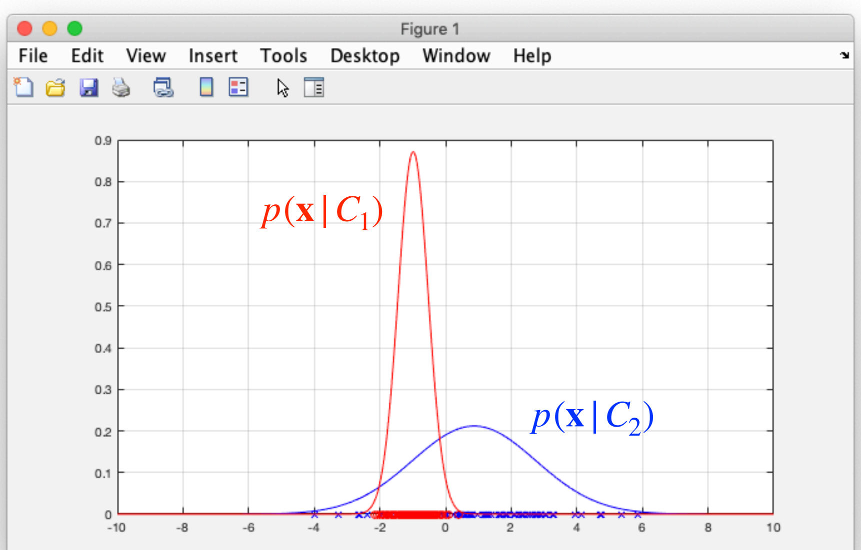
densidad de probabilidad
de $\mathbf{x}$, sea de la clase que sea

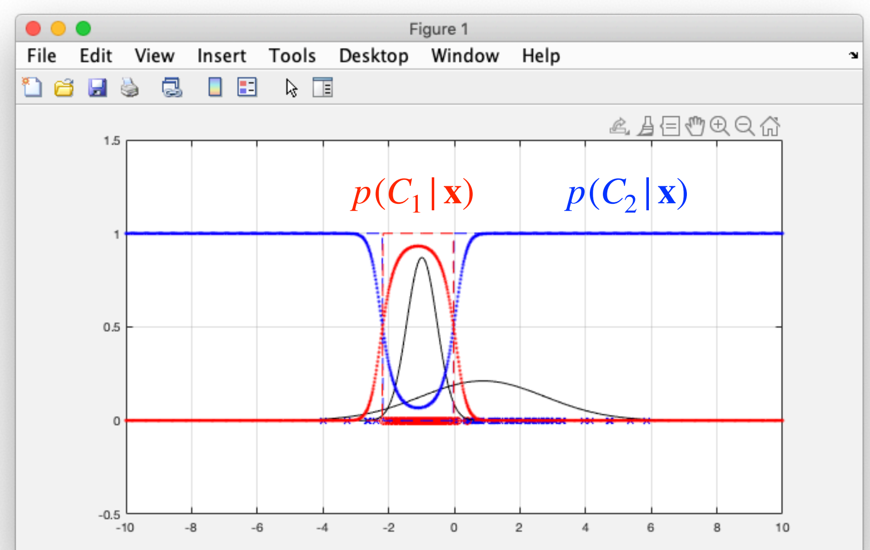
Clasificador Naïve Bayes

ejemplo en Matlab

```
% TRAINING DATA
n1 = 100;
n2 = 200;
p1 = randn(n1,1)*2+1;
p2 = randn(n2,1)*0.5-1;
p=[p1;p2];

% PRIOR PROBABILITIES FOR CLASS C1, C2
PC1 = n1/(n1+n2);
PC2 = n2/(n1+n2);

% TEST DATA
x=linspace(-10,10,1000);

% ESTIMATE THE PARAMETERS FOR THE DISTRIBUTIONS (suppose normally destributed)
mu1e = mean(p1);
mu2e = mean(p2);
sigma1e = std(p1);
sigma2e = std(p2);

% OBTAIN THE CONDITIONAL PROBABILITY DENSITIES FOR EACH CLASS P(X|Ci)
pxC1=normpdf(x,mu1e,sigma1e);
pxC2=normpdf(x,mu2e,sigma2e);

% TOTAL PROBABILITY DENSITY FUNCTION
px=pxC1*PC1+pxC2*PC2;

% APPLY BAYES RULE TO OBTAIN POSTERIOR PDF p(Ci|x)
PC1x=pxC1*PC1./px;
PC2x=pxC2*PC2./px;
```

```
% PLOT RESULTS
figure;

% training data
plot(p1,zeros(n1,1),'xb',p2,zeros(n2,1),'or')
title('training data');
axis([-10 10 -0.5 1.5]);
hold on;

% conditioned pdf of each class p(x|Ci)
plot(x,pxC1,'b');plot(x,pxC2,'r');
title('conditioned pdf of each class p(x|Ci)');

% posterior pdf's p(Ci|x)
plot(x,PC1x,'.b');plot(x,PC2x,'.r');
title('posterior pdfs p(Ci|x)');

% classification: assign the class with largest posterior probability
plot(x,PC1x>PC2x,'--b');plot(x,PC2x>PC1x,'--r');
title('classification: assign the class with largest posterior probability');
```





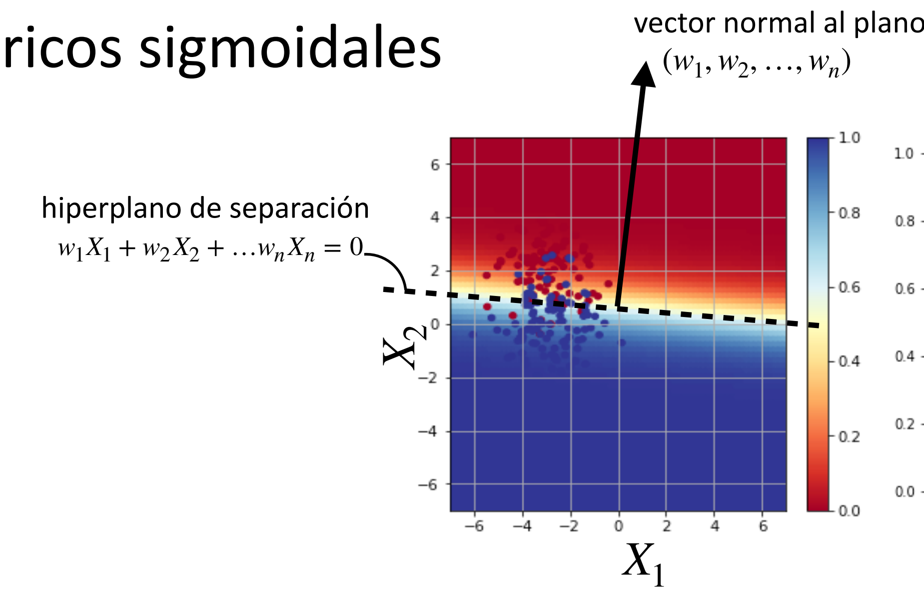
Clasificación

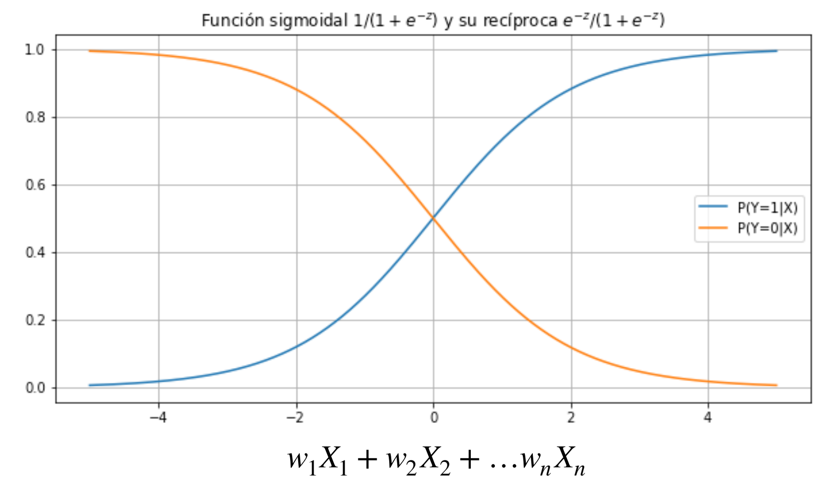
Regresión Logística

Suponemos modelos paramétricos sigmoidales

$$P(Y=0\,|\,X) = \frac{e^{-\sum_i w_i \cdot X_i}}{1+e^{-\sum_i w_i \cdot X_i}}$$

$$P(Y=1\,|\,X) = \frac{1}{1+e^{-\sum_i w_i \cdot X_i}}$$





La regla de clasificación sería

$$1 < \frac{P(Y=0\,|\,X)}{P(Y=1\,|\,X)} \quad \leftrightarrow \quad 0 < -\sum_i w_i X_i$$

Si tenemos L observaciones, $X_1, \ldots X_L$,
la *verosimilitud* de acuerdo con el modelo paramétrico $P(Y\,|\,X, w)$ sería

$$\prod_l P(Y_l\,|\,X_l, w)$$

notación para L muestras

componente

X_{li}

observación (muestra)

Clasificación

Regresión Logística

Buscamos los pesos w del modelo que maximicen la verosimilitud

$$w \leftarrow \arg\max_w \prod_l P(Y_l | X_l, w)$$

log-likelihood

Tomando logaritmos, se convierten en sumas

$$w \leftarrow \arg\max_w \sum_l \ln P(Y_l | X_l, w) \quad \text{o bien} \quad w \leftarrow \arg\max_w L(w)$$

Y_l solo toma dos valores $\{0,1\}$, podemos separar sumandos en dos: $Y_l = 1,\ Y_l = 0$

$$L(w) = \sum_l \left\{ Y_l \ln P(Y_l = 1 | X_l, w) + (1 - Y_l) \ln P(Y_l = 0 | X_l, w) \right\}$$

$$L(w) = \sum_l \left\{ Y_l \ln \frac{P(Y_l = 1 | X_l, w)}{P(Y_l = 0 | X_l, w)} + \ln P(Y_l = 0 | X_l, w) \right\}$$

$$L(w) = \sum_l \left\{ Y_l \left(\sum_i^n w_i X_{li} \right) + \ln \frac{e^{-\sum_i w_i X_{li}}}{1 + e^{-\sum_i w_i X_{li}}} \right\}$$

Clasificación

Regresión Logística

derivando respecto a los pesos w_i obtenemos el gradiente

$$\frac{\partial L(w)}{\partial w_i} = \sum_l X_{li}(Y_l - P(Y_l = 1 | X_l, w))$$

… y del gradiente la regla de aprendizaje

$$w_i \leftarrow w_i + \nu \sum_l X_{li}(Y_l - P(Y_l = 1 | X_l, w))$$

ejemplo programándolo (¡muy sencillo!)

```
# Clase para iteración de regresión logística
# ¡¡La solución es muy simple!!

class LogisticRegression():
    def fit(self, X, y, n_iter=4000, lr=0.01):
        self.w = np.random.rand(X.shape[1])
        for _ in range(n_iter):
            self.w -= lr * (self.predict(X) - y).dot(X)
    def predict(self, X):
        return sigmoid(X.dot(self.w))
```

```
# Instanciamos nuestra clase LogisticTegression
a = LogisticRegression()

# ejecutamos el método fit para entrenar
a.fit(X,y,lr=0.0001)

# Definimos datos de test (Xt)

# ejecutamos el método predict()
yt = a.predict(Xt)
```

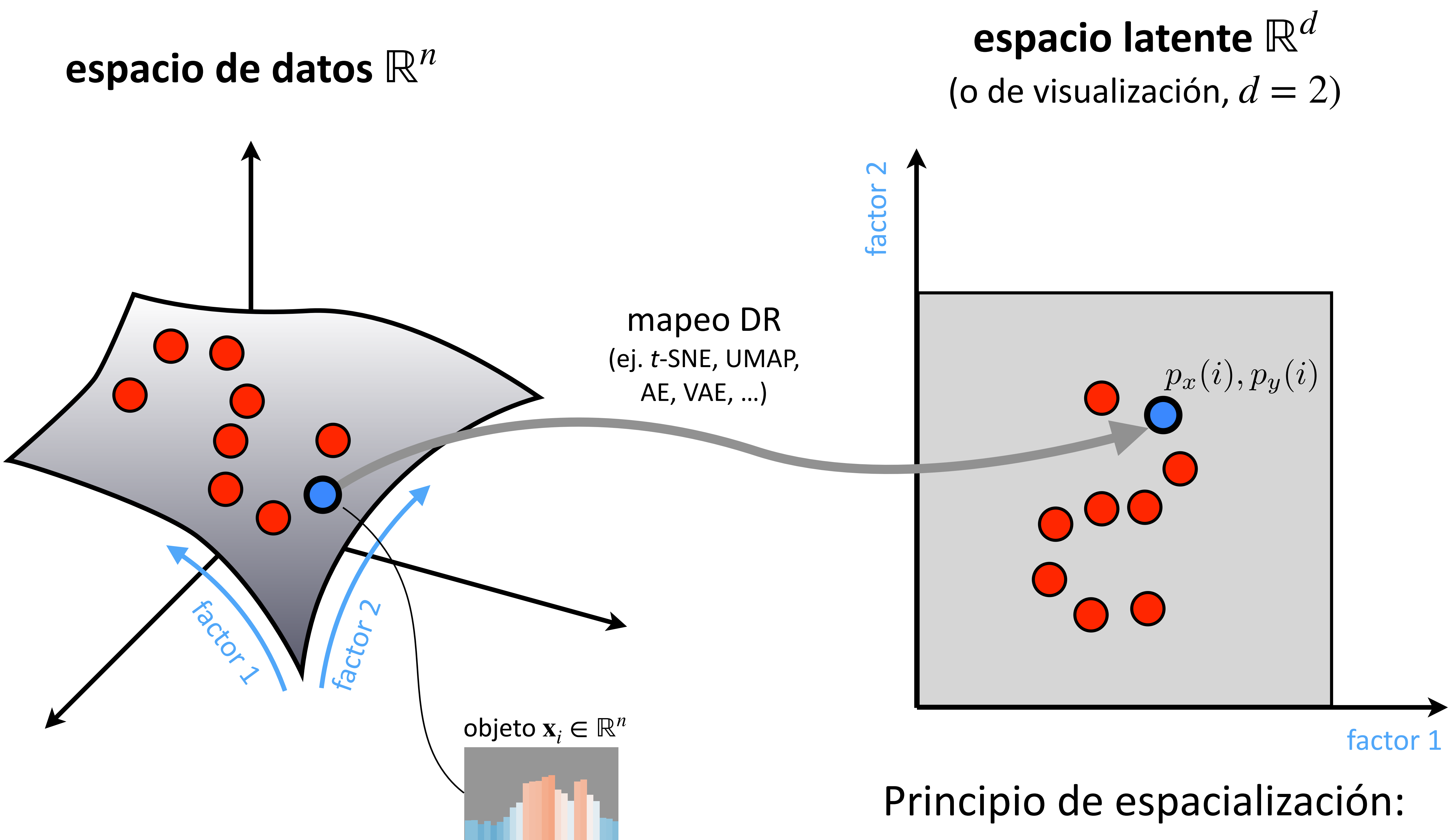
ejemplo con *scikit-learn*

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

X, y = load_iris(return_X_y=True)

clf = LogisticRegression(random_state=0).fit(X, y)

clf.predict(X[:2, :])
clf.predict_proba(X[:2, :])
clf.score(X, y)
```

Reducción de la Dimensionalidad

Reducción de la dimensionalidad

preservación de similitudes



Principio de espacialización:
"parecido" ↔ "cercano"

Reducción de la dimensionalidad

ejemplo en scikit-learn


```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names
```

proyecciones lineales

PCA (*Principal Component Analysis)*

```
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)
```

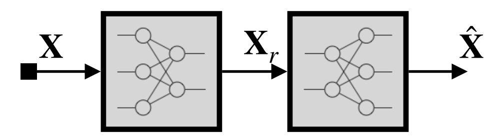
manifold learning

UMAP, *t*-SNE, …

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2,perplexity=20)
X_r = tsne.fit_transform(X)
```

autoencoders

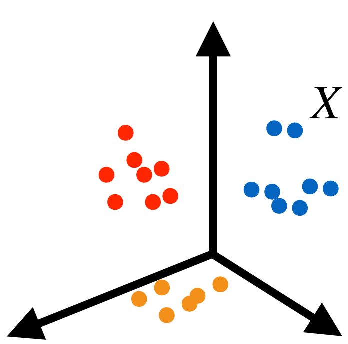
AE, deep AE, …



$X \in \mathbb{R}^{n,4}$

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
..	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

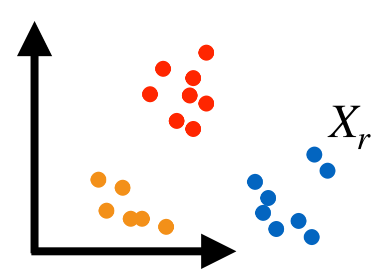
[150 rows x 4 columns]



$X_r \in \mathbb{R}^{n,2}$

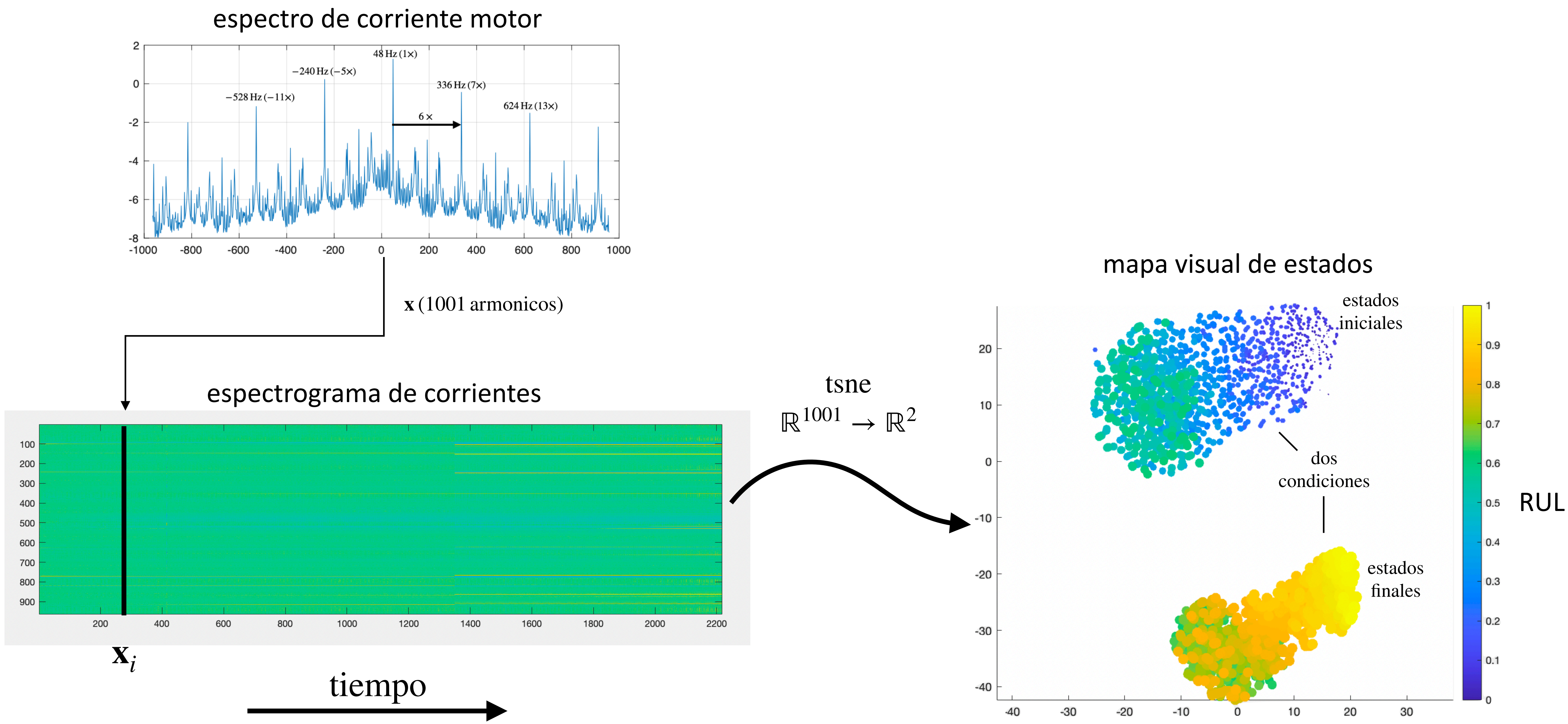
	componente 1	componente 2
0	-2.684126	0.319397
1	-2.714142	-0.177001
2	-2.888991	-0.144949
3	-2.745343	-0.318299
4	-2.728717	0.326755
..	...	...
145	1.944110	0.187532
146	1.527167	-0.375317
147	1.764346	0.078859
148	1.900942	0.116628
149	1.390189	-0.282661

[150 rows x 2 columns]



Reducción de la dimensionalidad

ejemplo estados motor de inducción



Aplicaciones

Machine Learning

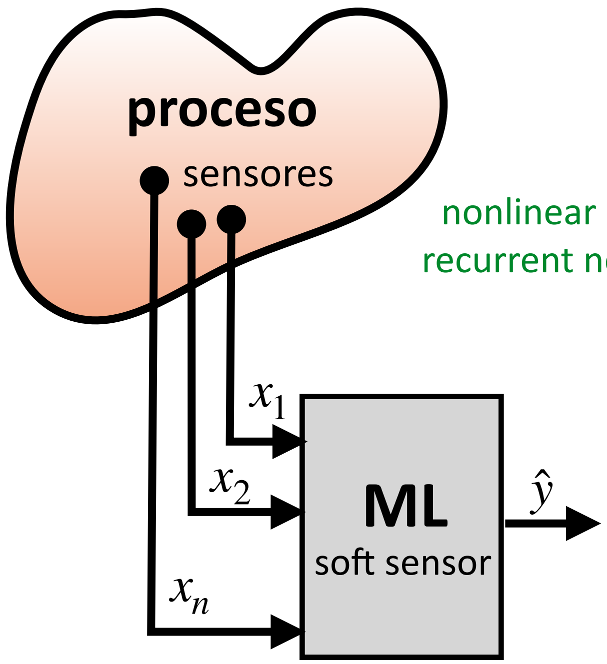
tareas más frecuentes

Universidad de Oviedo

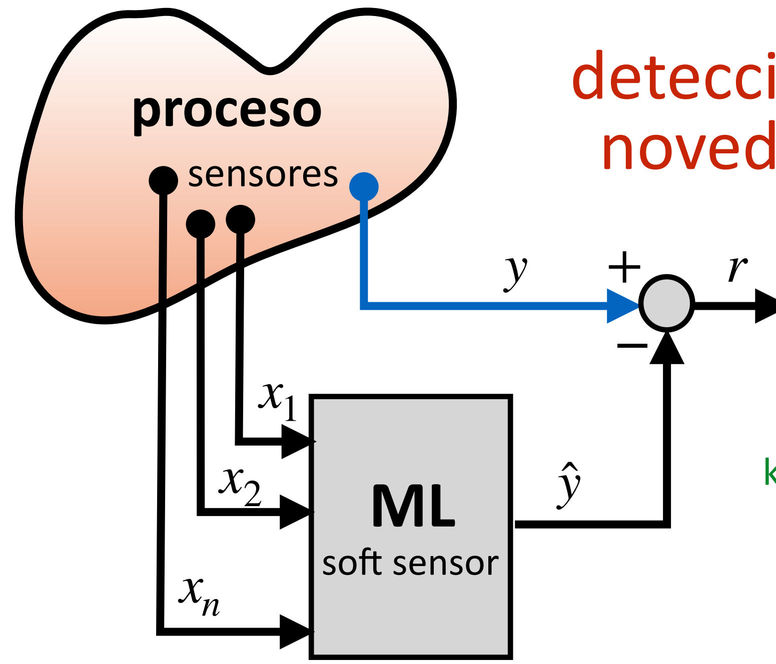
INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

estimación

deep learning (CNN)
nonlinear dynamic regression (NARX, SINDy)
recurrent networks (echo-state networks, ESN)



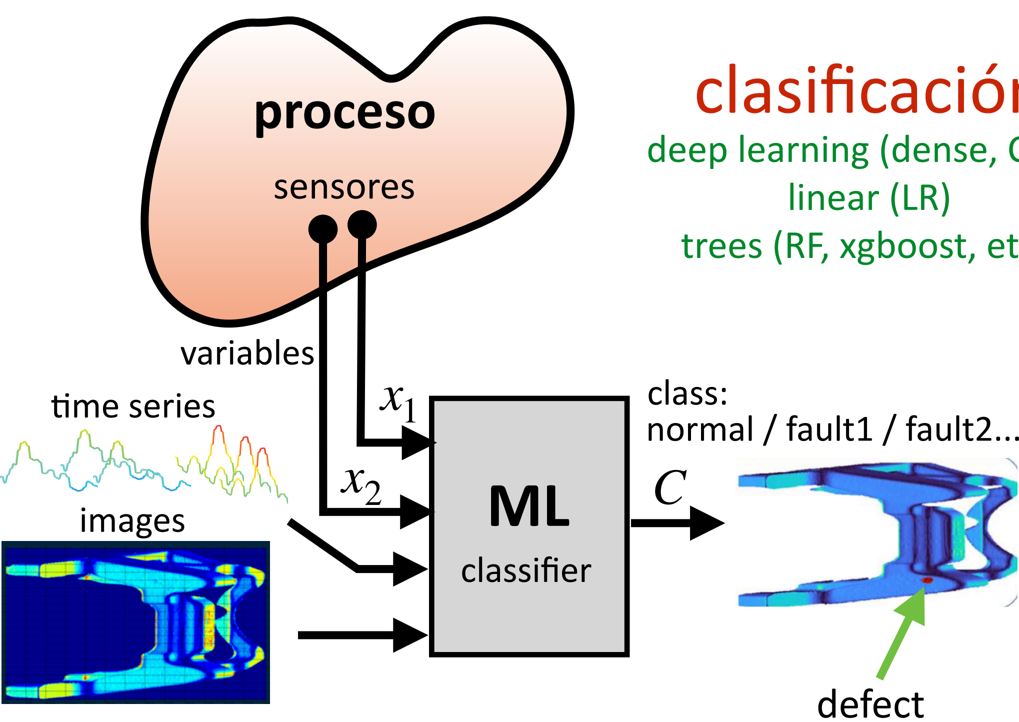
detección de novedades



kernel methods (KDE, SVR)
deep learning (AE, VAE)
analytical redundancy

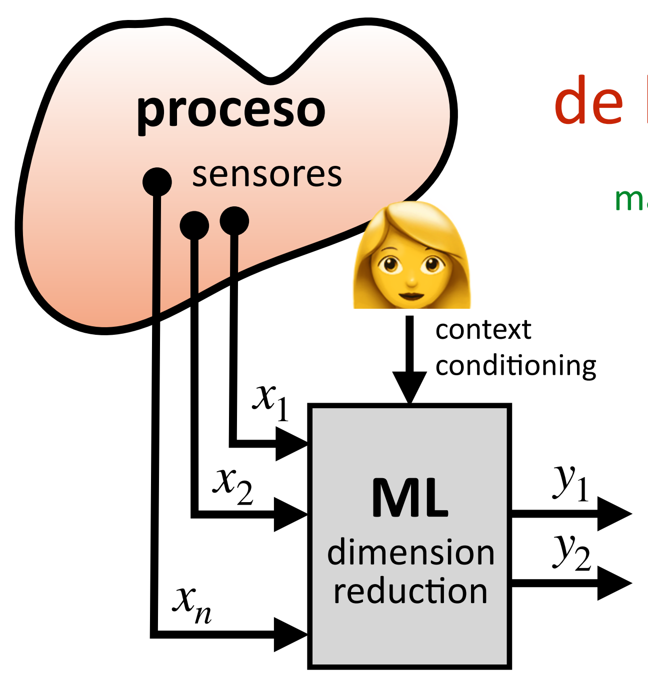
clasificación

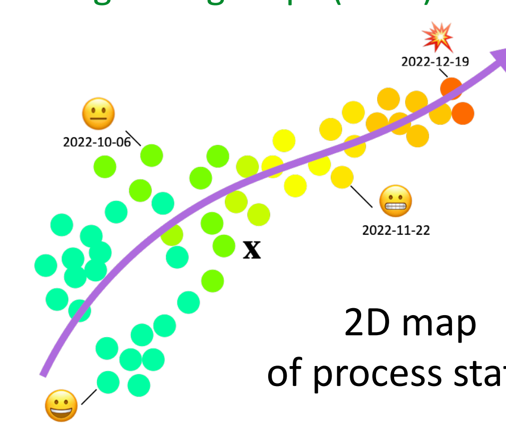
deep learning (dense, CNN)
linear (LR)
trees (RF, xgboost, etc.)



reducción de la dimensionalidad

manifold learning (tSNE, UMAP)
deep learning (AE, VAE)
linear (PCA)
self-organizing maps (SOM)

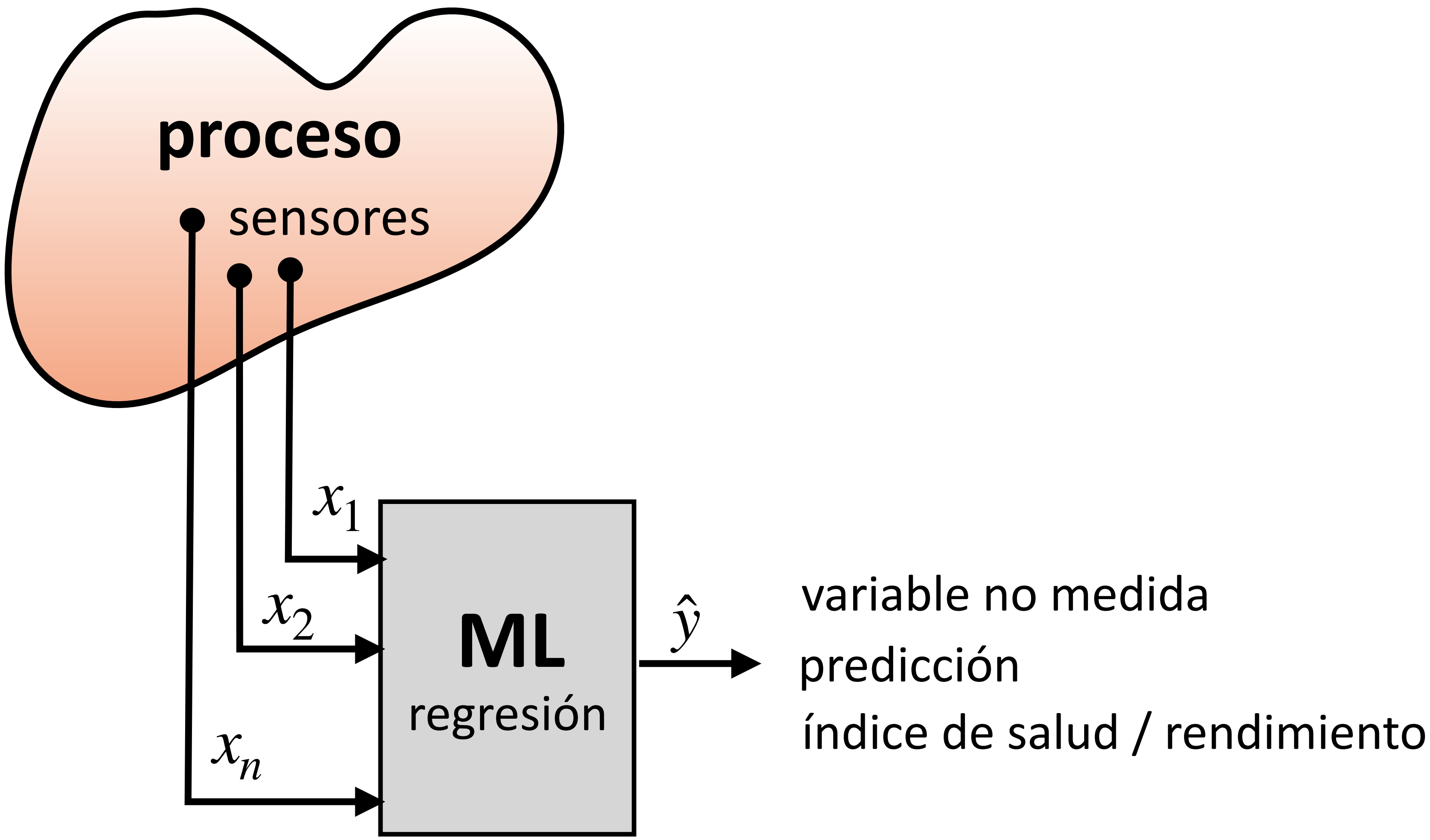




2D map
of process states

Sensores virtuales (soft sensors)

Estimación - Predicción - Índices de rendimiento

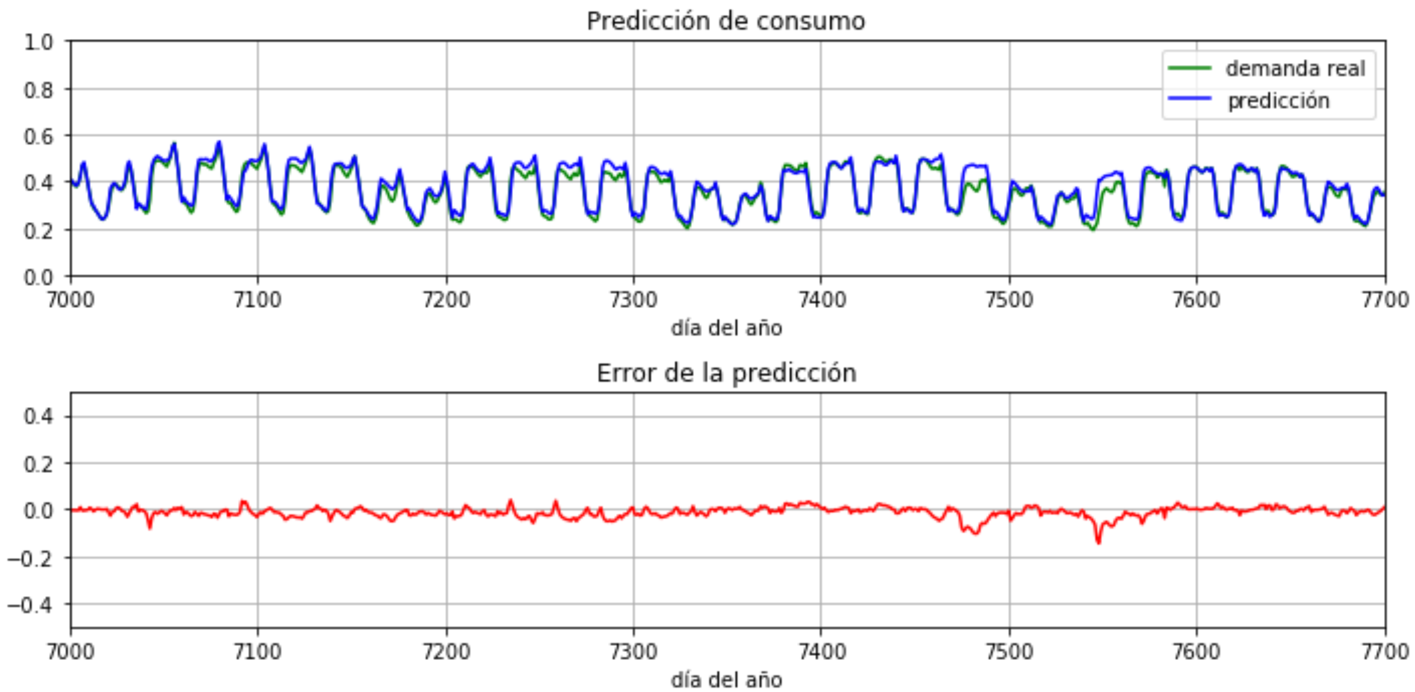


Sensores virtuales (soft sensors)

Predicción

Universidad de Oviedo

Predicción de demanda eléctrica



Extracción de características

```python
# CREAMOS REGRESOR DINÁMICO (TAP-DELAY) DE DATOS DE ENTRENAMIENTO
X = []
y = []

# entrenamos con los datos de un año (8760 registros horarios)
N = 8760

# estimamos los datos de varias horas después (ej. ahead = 24 horas)
ahead = 24
for k in range(169,N):
    X.append(np.hstack((
                C[k-24:k-1],            # vector de consumo diario (24h)
                W[k],                   # semana del año
                D[k],                   # día de la semana
                H[k],                   # hora del día
                1                       # término independiente
            )).tolist()  )
    y.append(C[k+ahead].tolist() )


# CREAMOS REGRESOR DINÁMICO (TAP-DELAY) DE DATOS DE TEST
Xtst = []
ytst = []
Ntst = 10000

for k in range(169,Ntst):
    Xtst.append(np.hstack((
                C[k-24:k-1],            # vector de consumo diario (24h)
                W[k],                   # semana del año
                D[k],                   # día de la semana
                H[k],                   # hora del día
                1                       # término independiente
            )).tolist()  )
    ytst.append(C[k+ahead].tolist() )
```

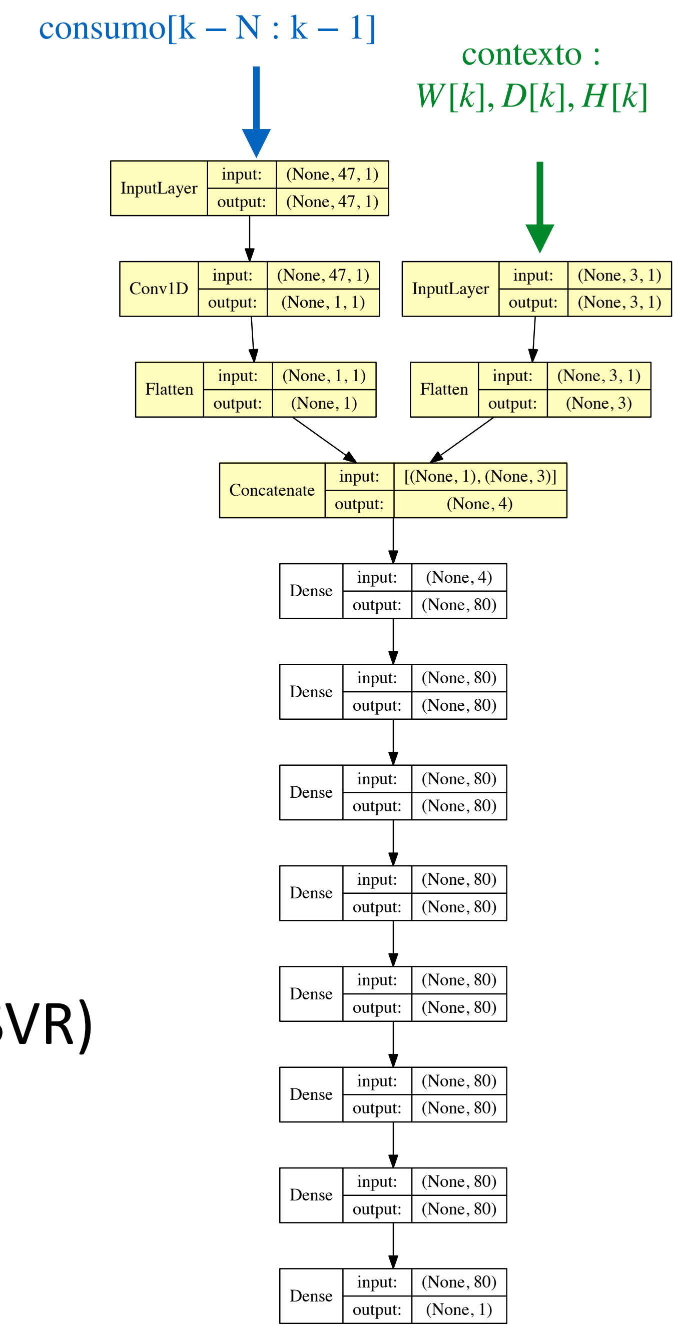
1. Regresión Lineal (Ridge)

```python
# PREDICCIÓN LINEAL
from sklearn import linear_model
clf = linear_model.Ridge (alpha = 0.1)
clf.fit(X_norm,y_norm)
yest_norm = clf.predict(Xtst_norm)
```

2. Regresión Vectores de Soporte (SVR)

```python
# PREDICCIÓN Support Vector Regression (SVR)
from sklearn.svm import SVR
import numpy as np

# este método permite calcular los coeficientes de una
aproximación rbf
# mediante una optimización "robusta". No está limitado a rbf. Es
extensible a otros tipos de aproximaciones.
clf = SVR(kernel='rbf', gamma=.5, C=10, epsilon=.02)

clf.fit(X_norm,y_norm)
yest_norm = clf.predict(Xtst_norm)
```

3. Red Convolucional (CNN)



(González, 2017)

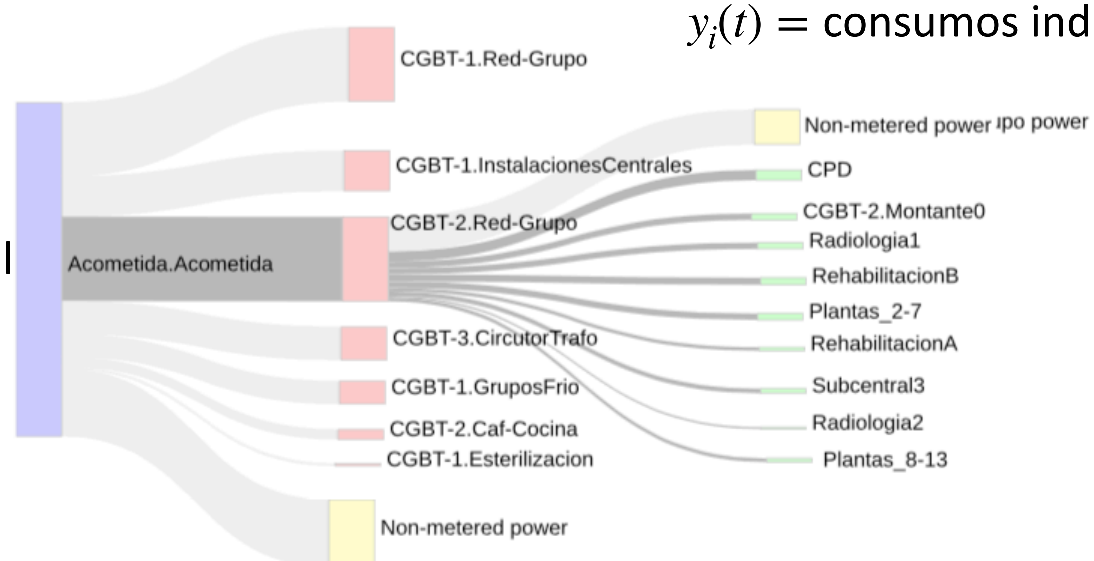
Sensores virtuales (soft sensors)

Estimación → NILM

(García et al, 2021)

$y_i(t)$ = consumos individuales

$x(t)$ = consumo global

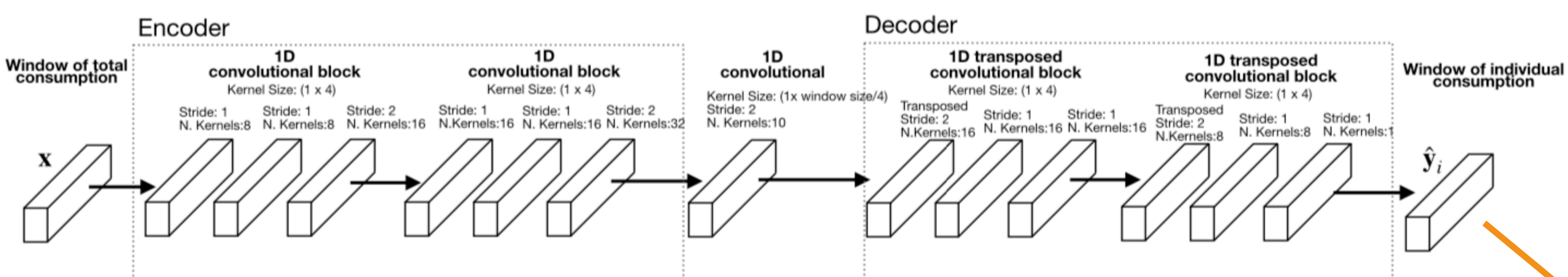


$$x(t) = y_1(t) + y_2(t) + \ldots + y_n(t) + e(t)$$

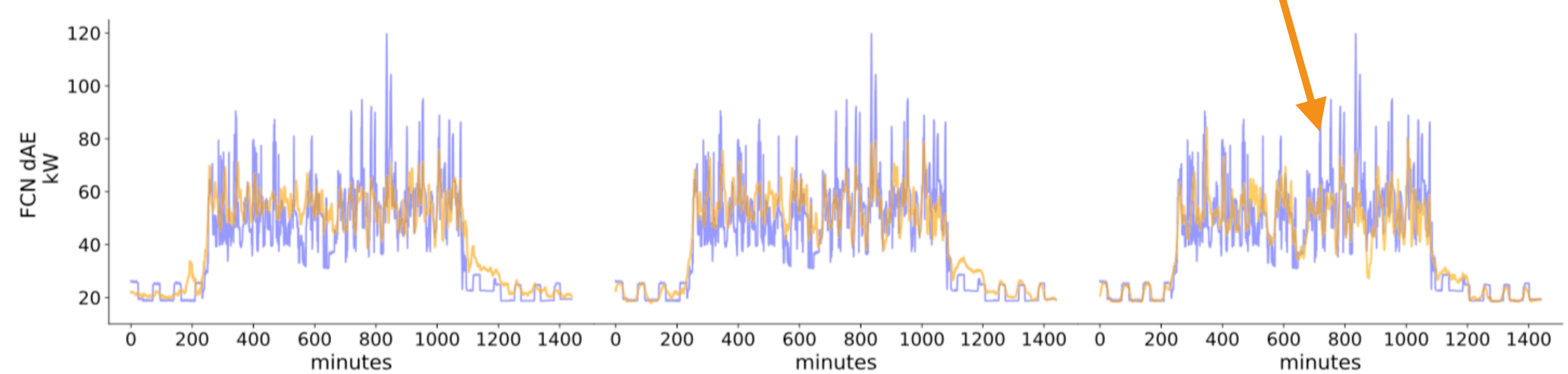
$$f_i : \mathbf{x} \rightarrow \mathbf{y}_i$$

Estimación seq2seq

serie temporal de consumos globales

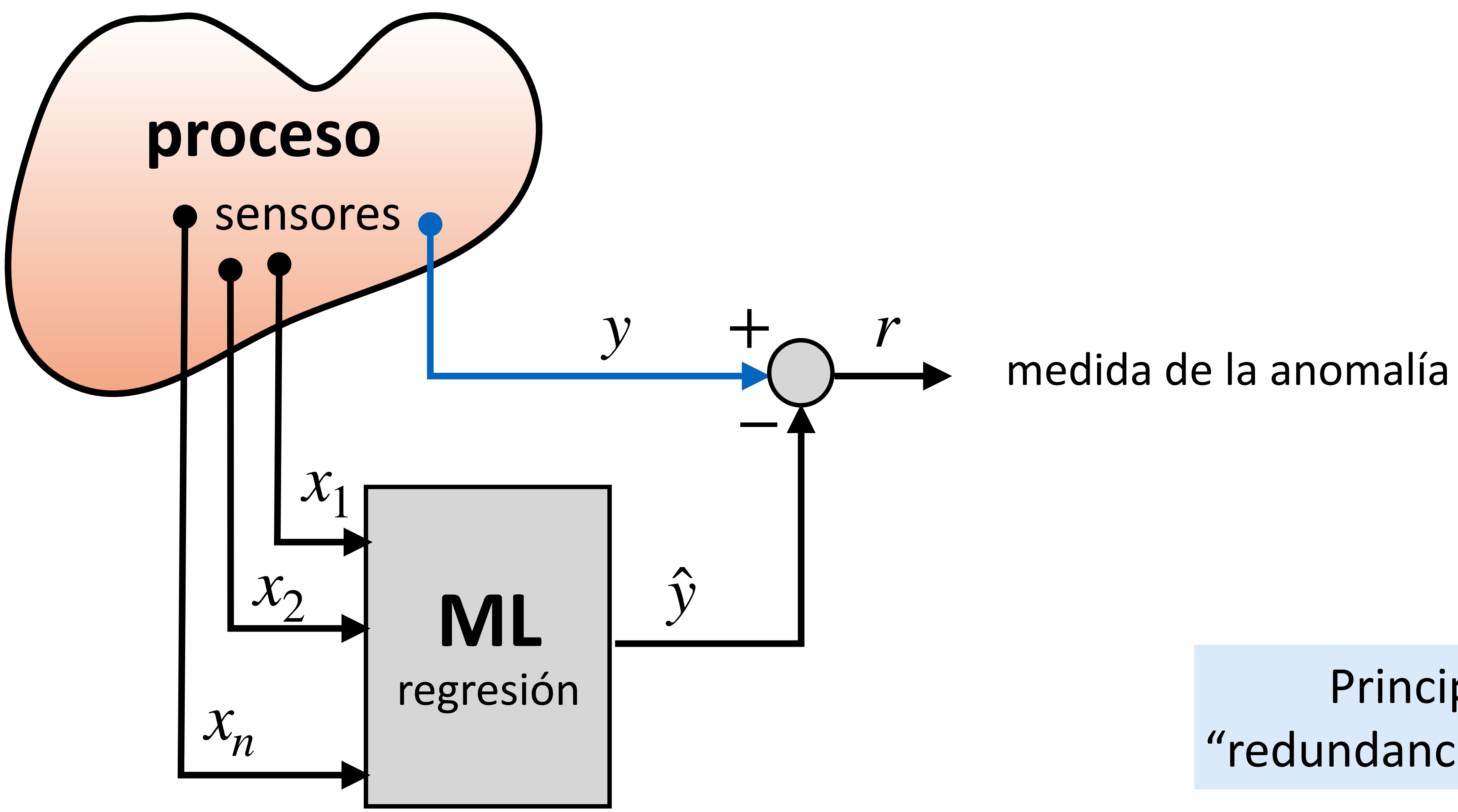


serie temporal de consumos individuales



Detección de Novedades

Detección de fallos / Anomalías

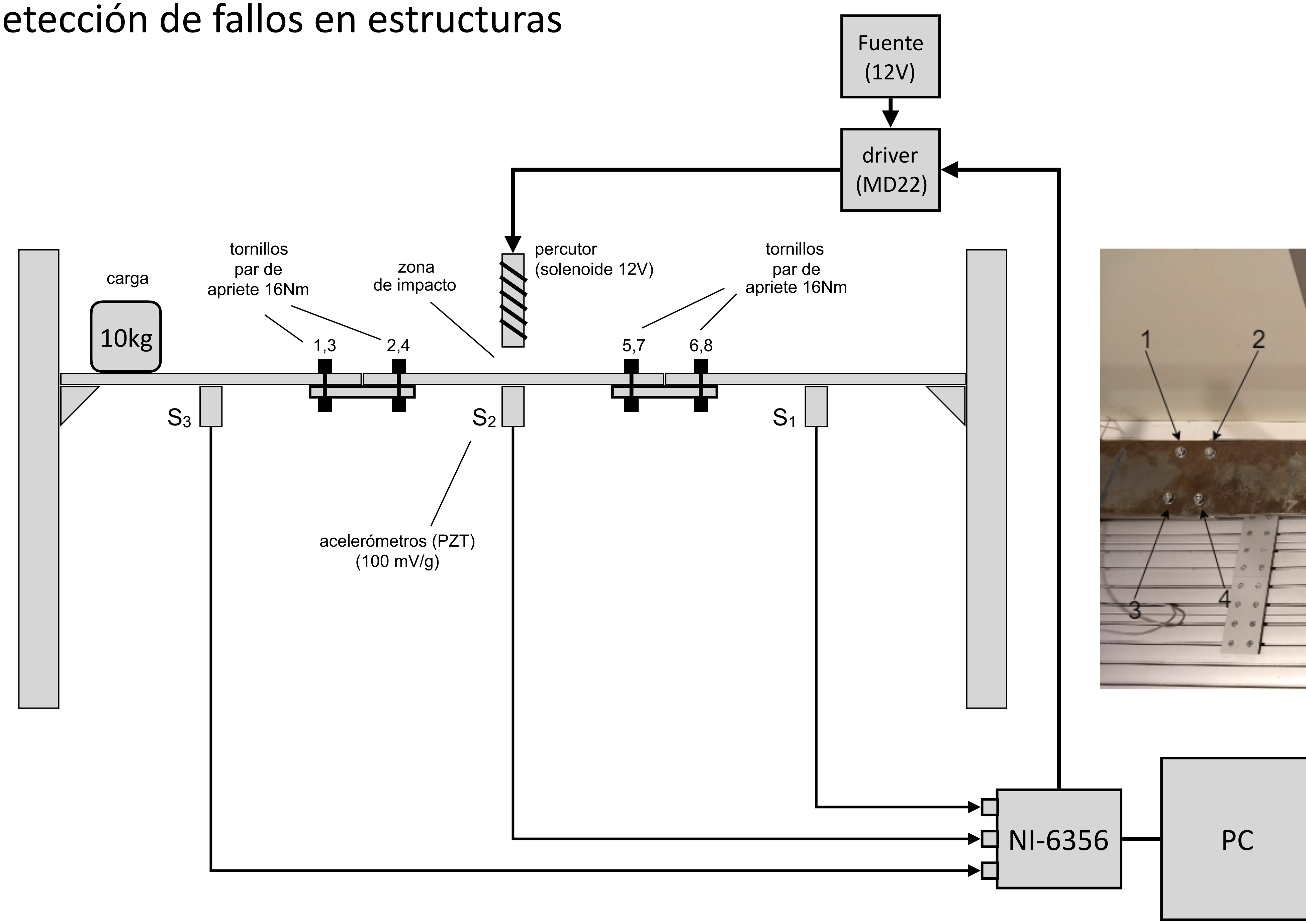


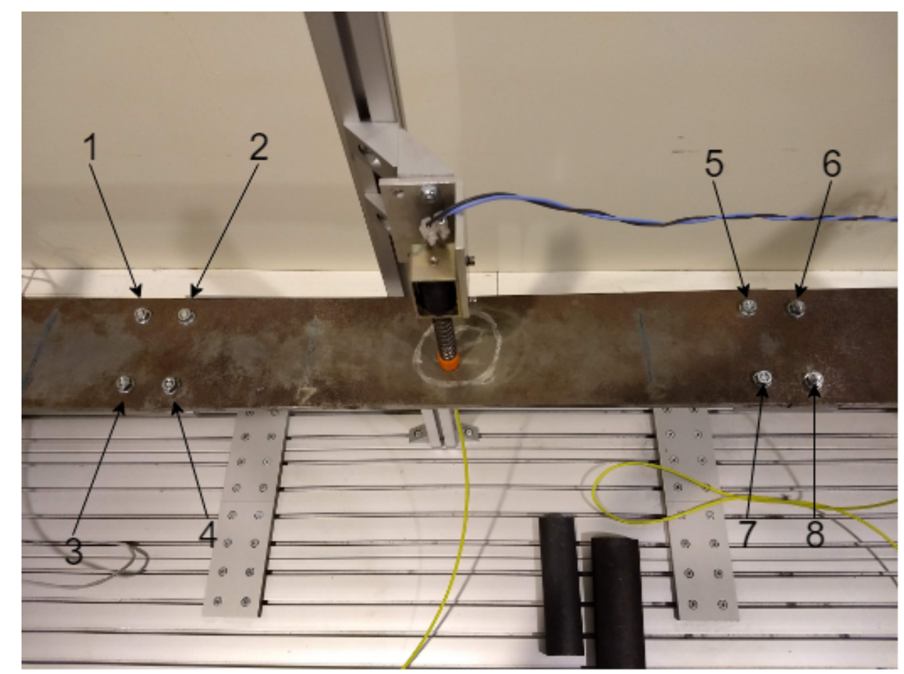
Principio de "redundancia analítica"

Detección de Novedades

Detección de fallos / Anomalías

Detección de fallos en estructuras

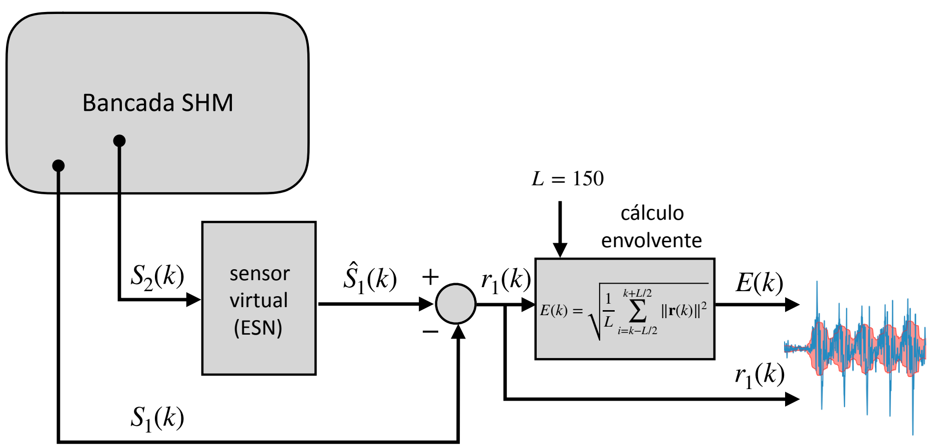


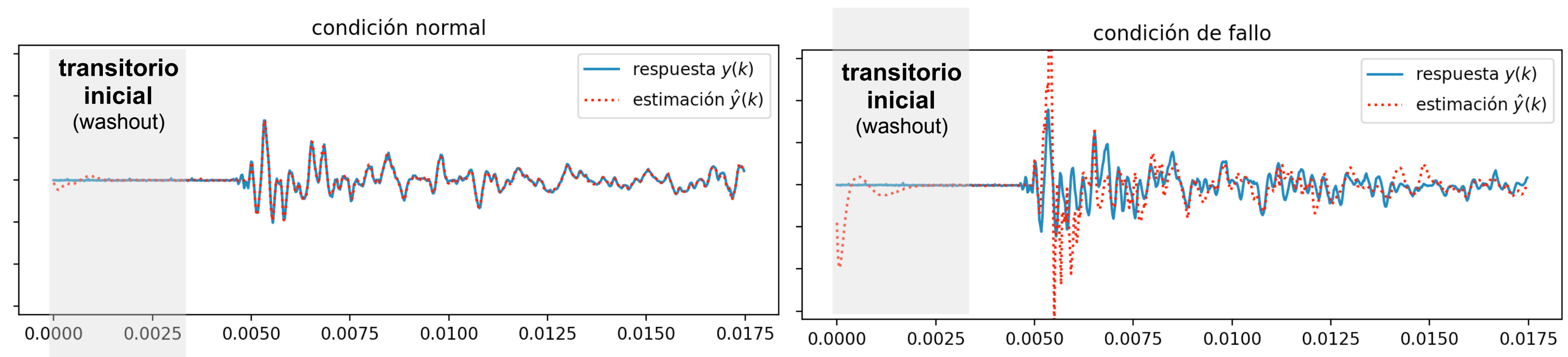


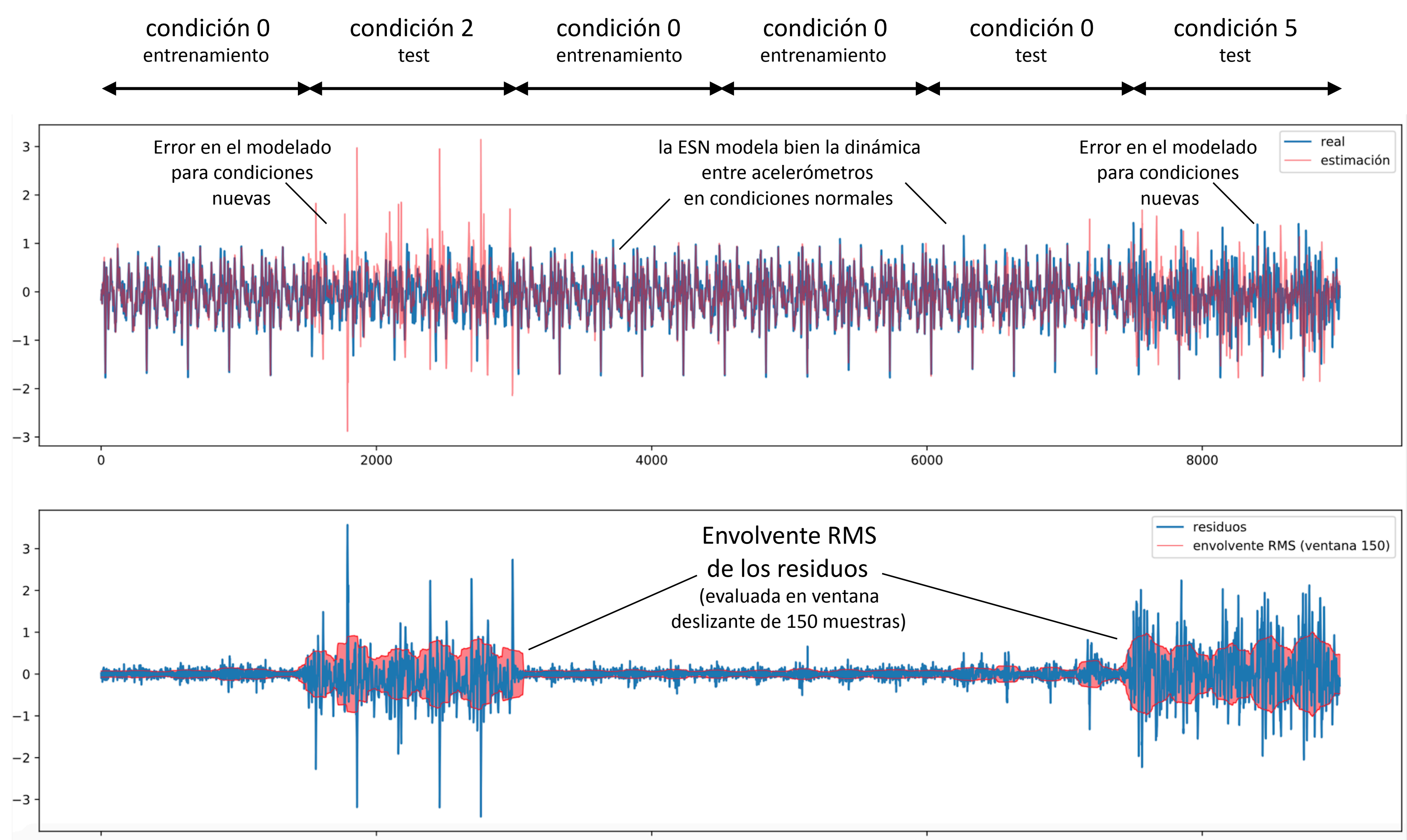
Detección de Novedades

Detección de fallos / Anomalías

Detección de fallos en estructuras





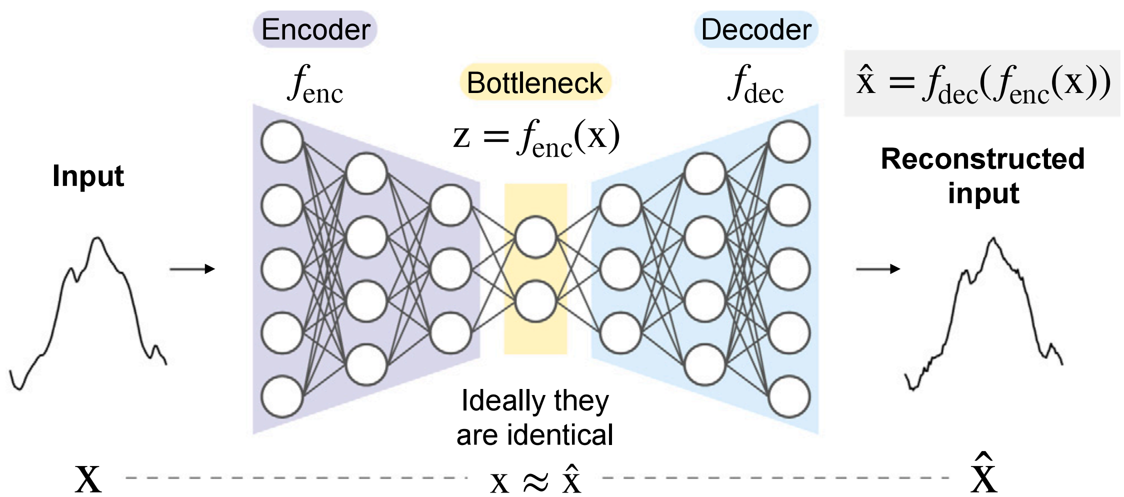


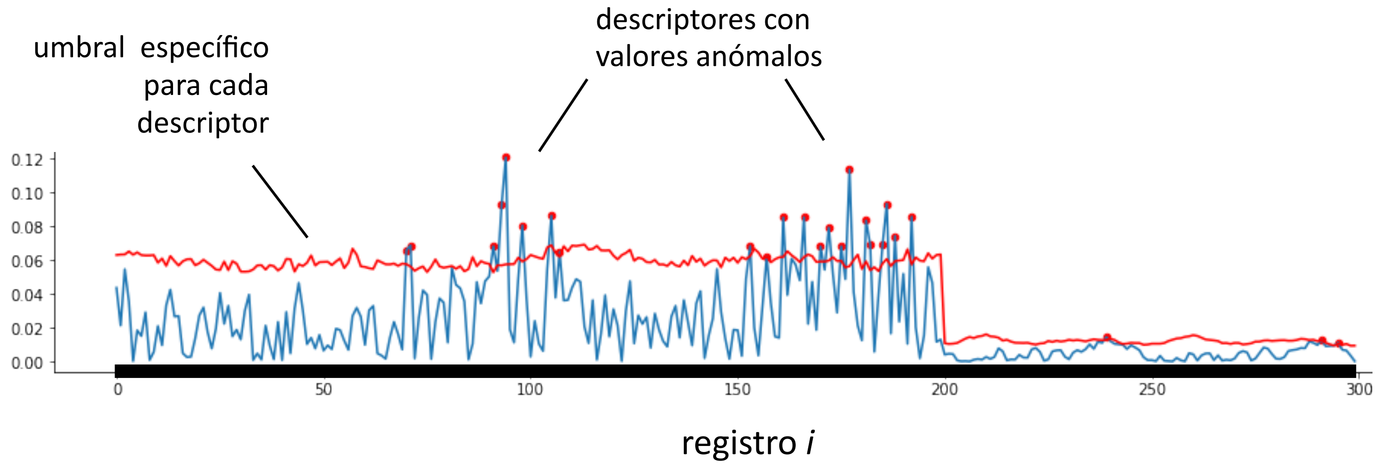
Detección de Novedades

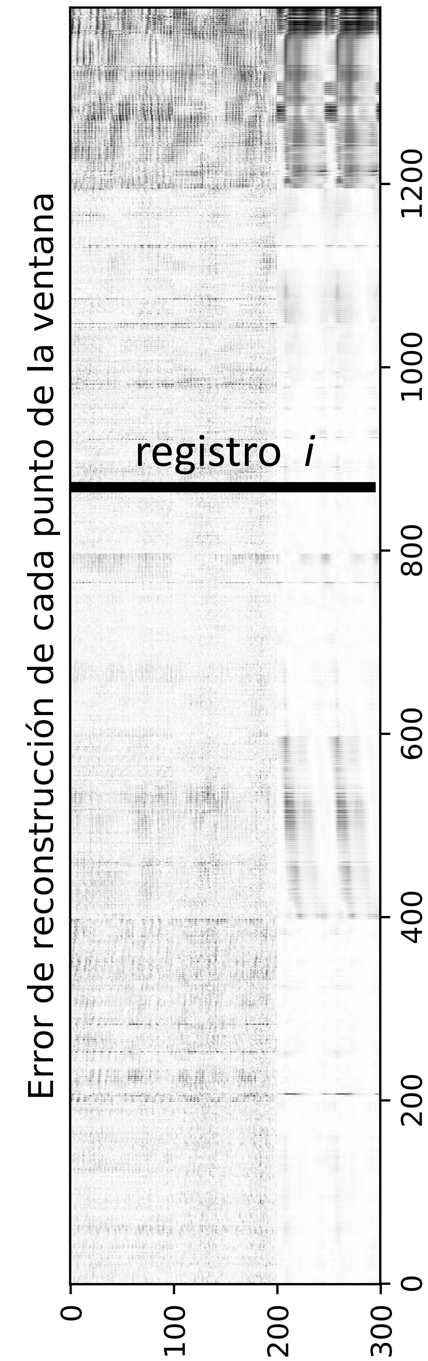
Detección de fallos / Anomalías

Anomalías en máquinas eléctricas

fuente: (González et al. 2022)





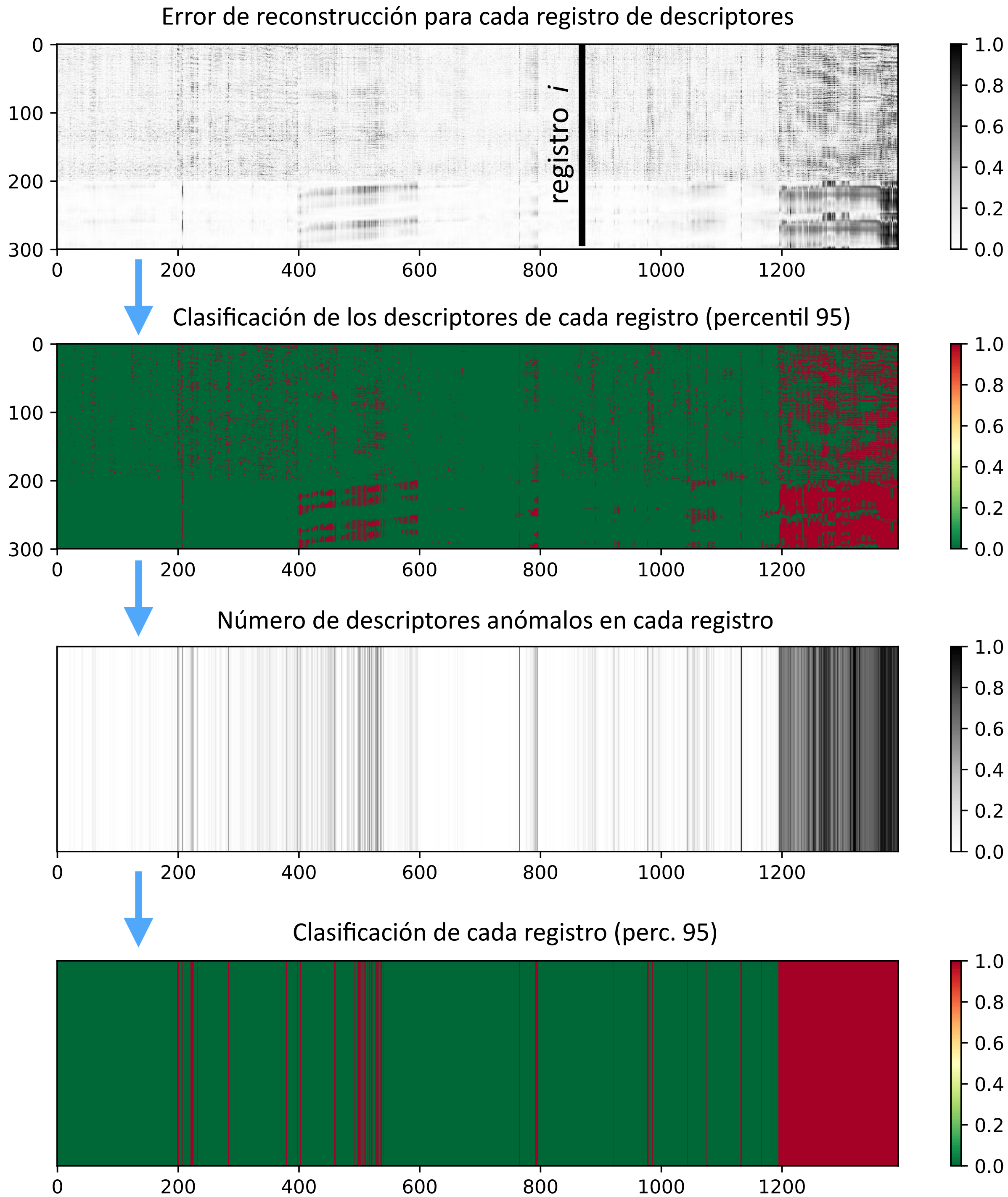


Detección de Novedades

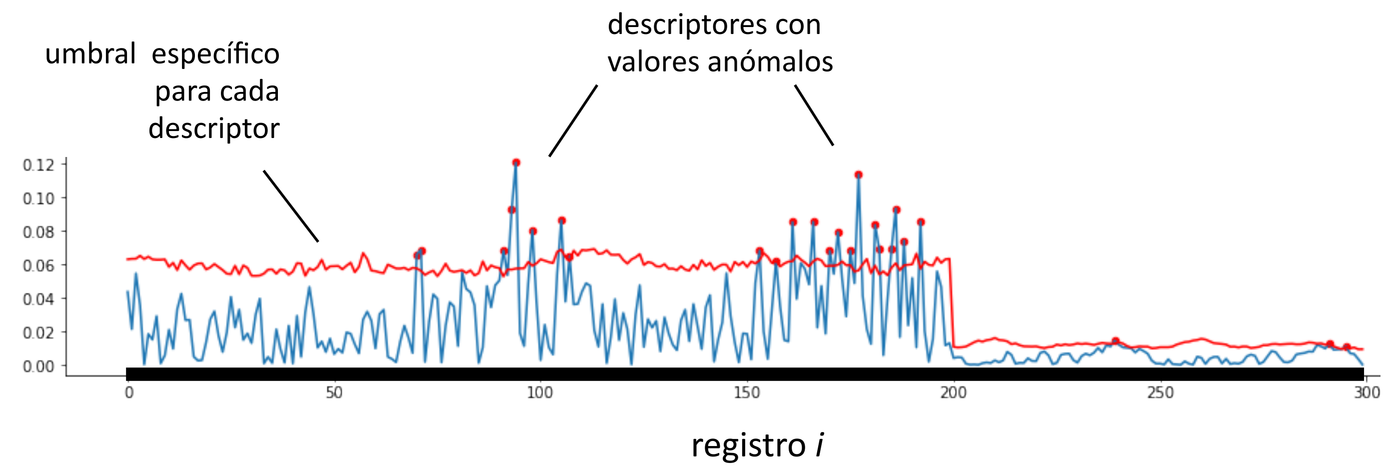
Detección de fallos / Anomalías

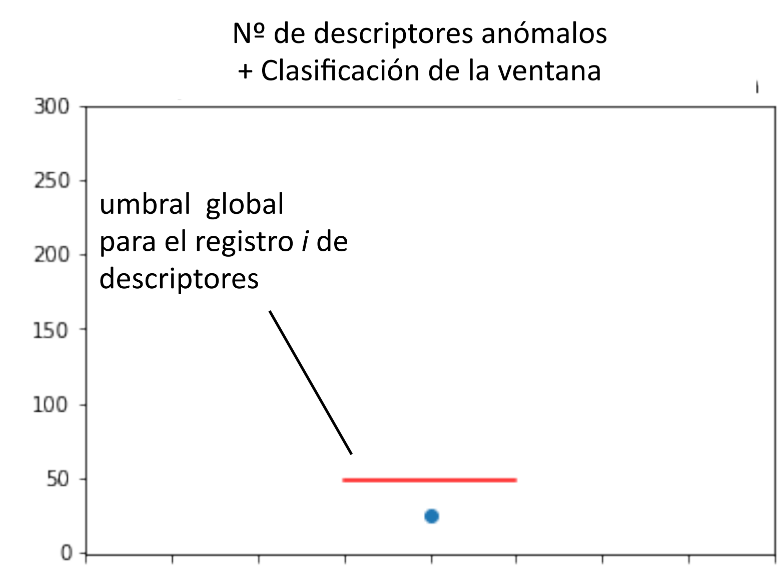
Universidad de Oviedo

1600 registros con 300 descriptores



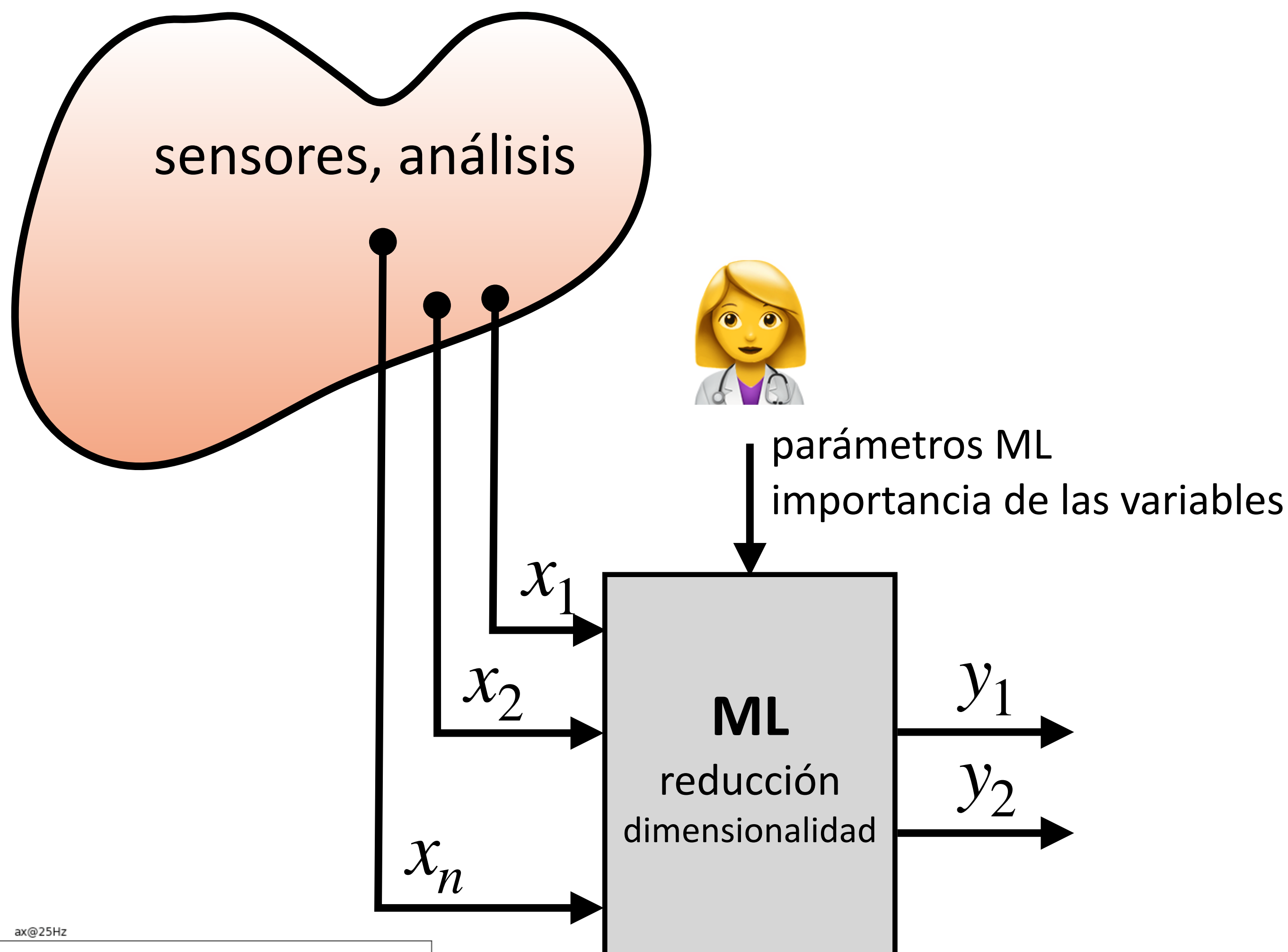
Análisis de anomalías del registro

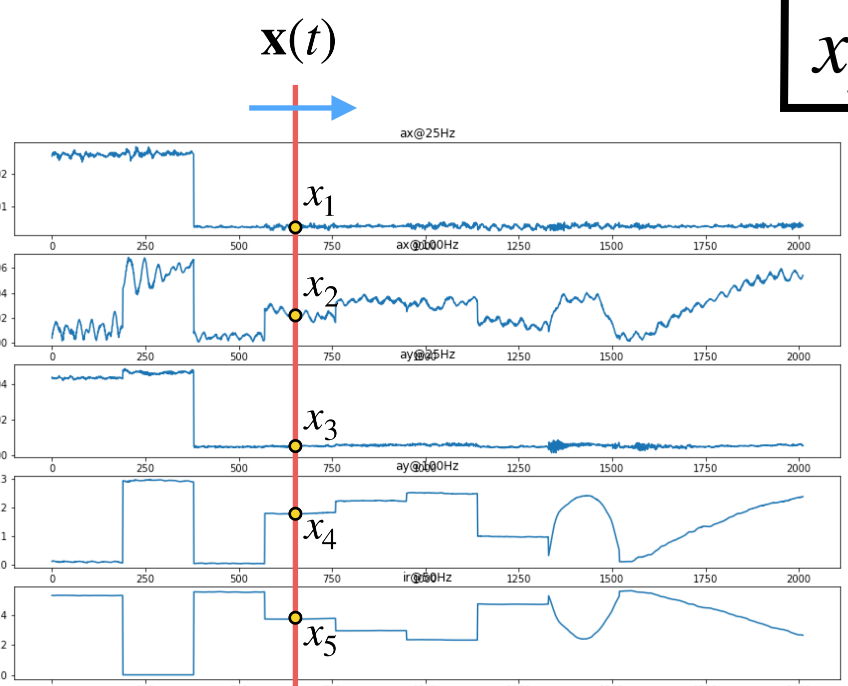


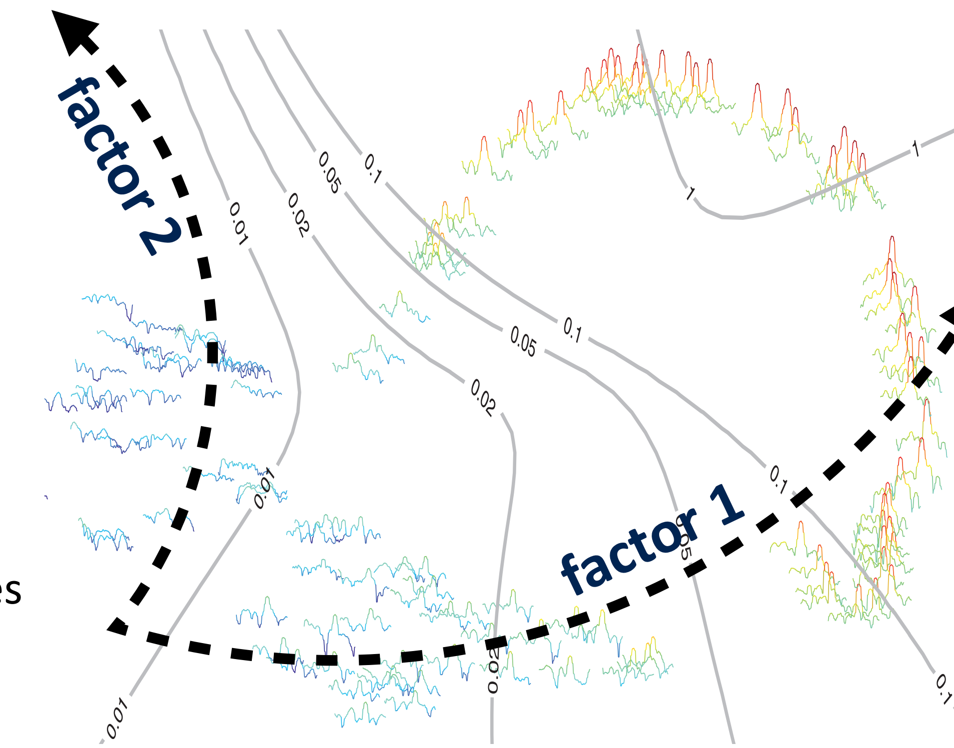


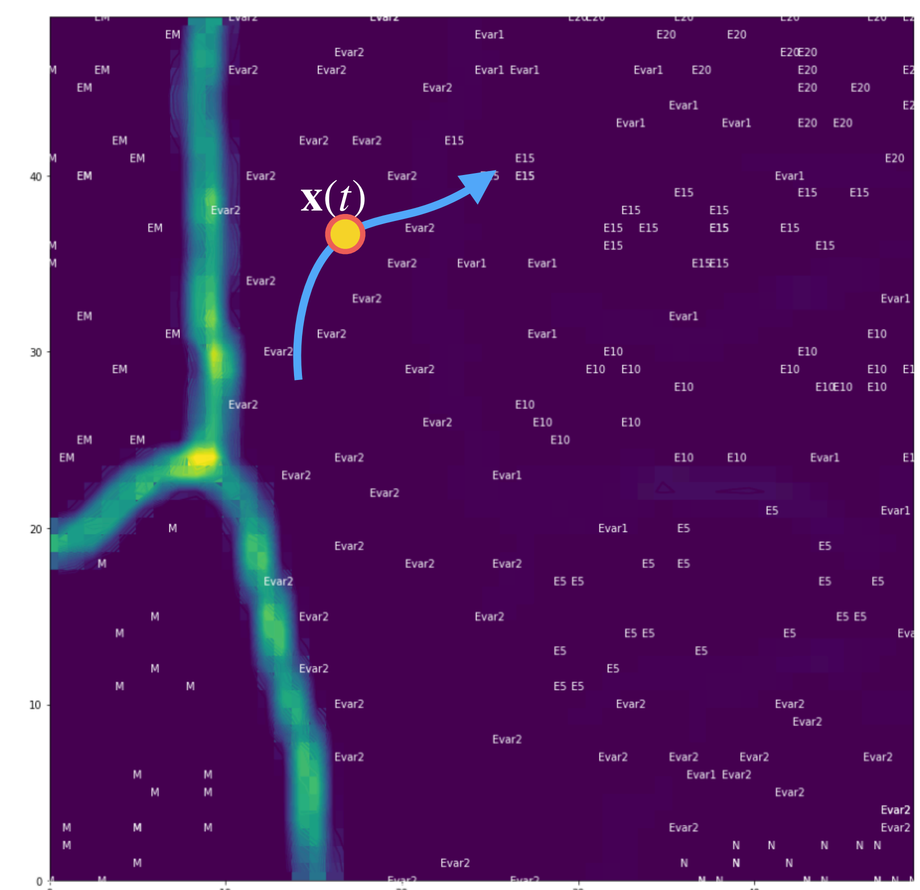
Machine learning

visualización









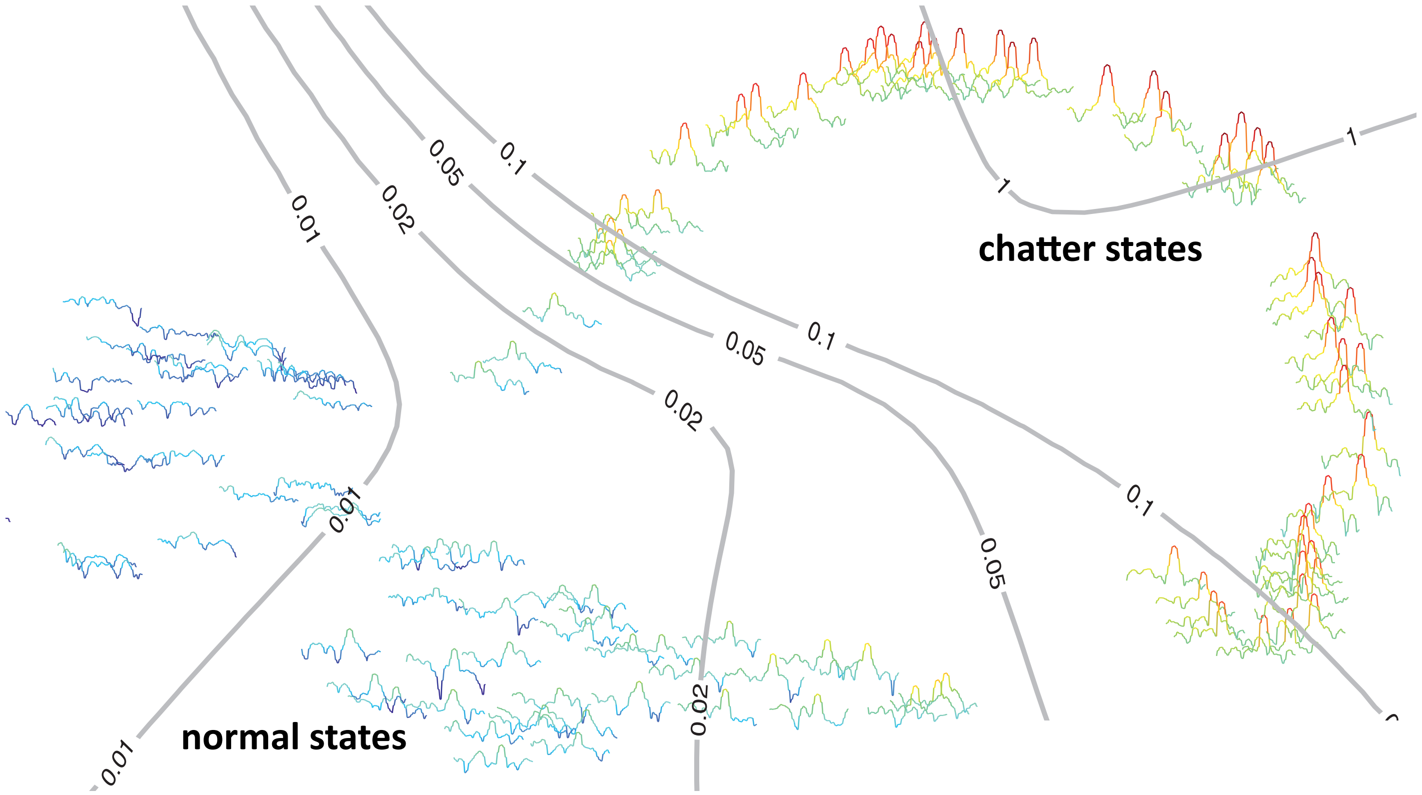
Visualización

representación 2D/3D
interacción, exploración,
animación

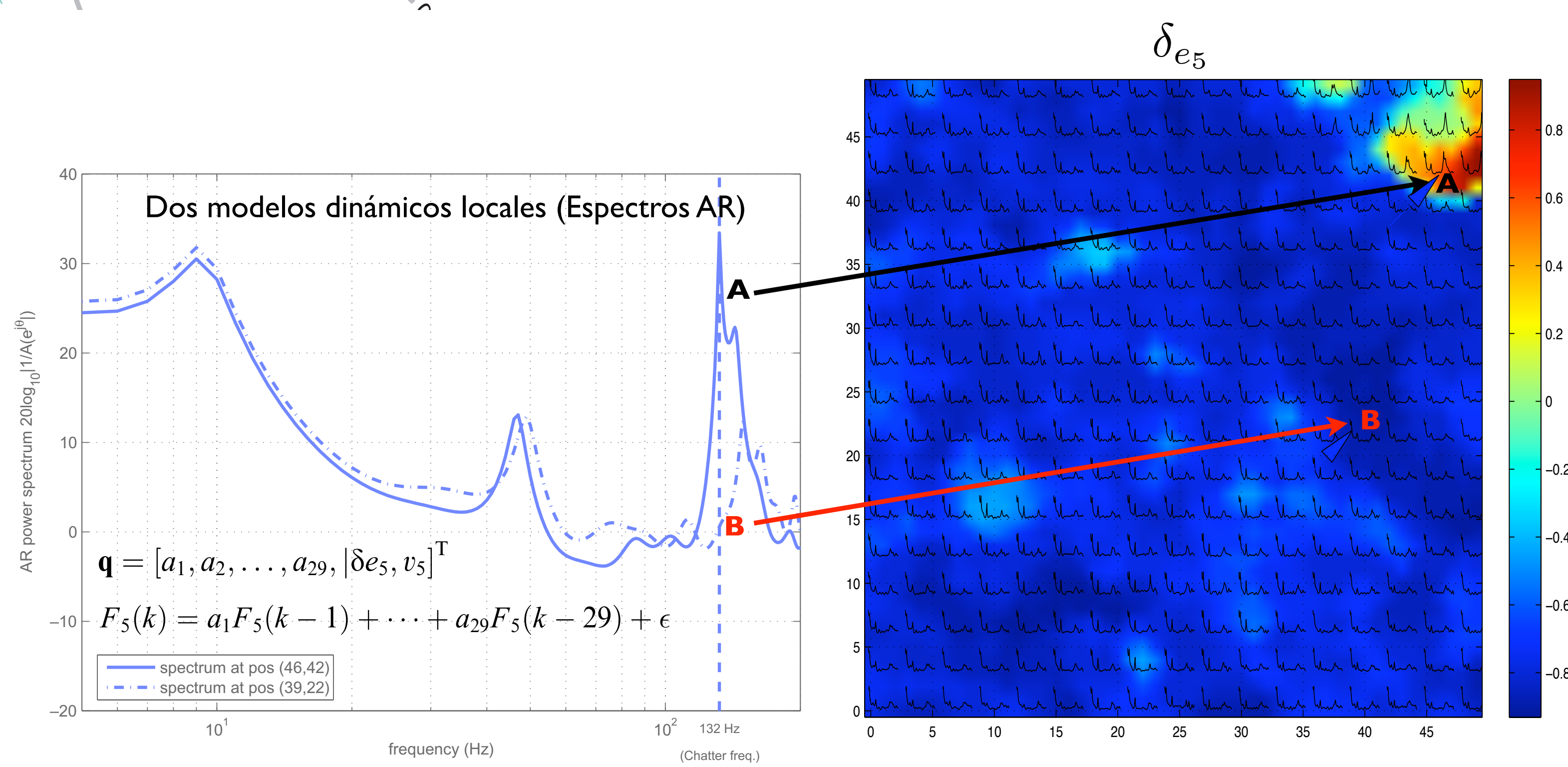
Reducción de la dimensionalidad

mapas de estados del proceso

Mapas de vibración



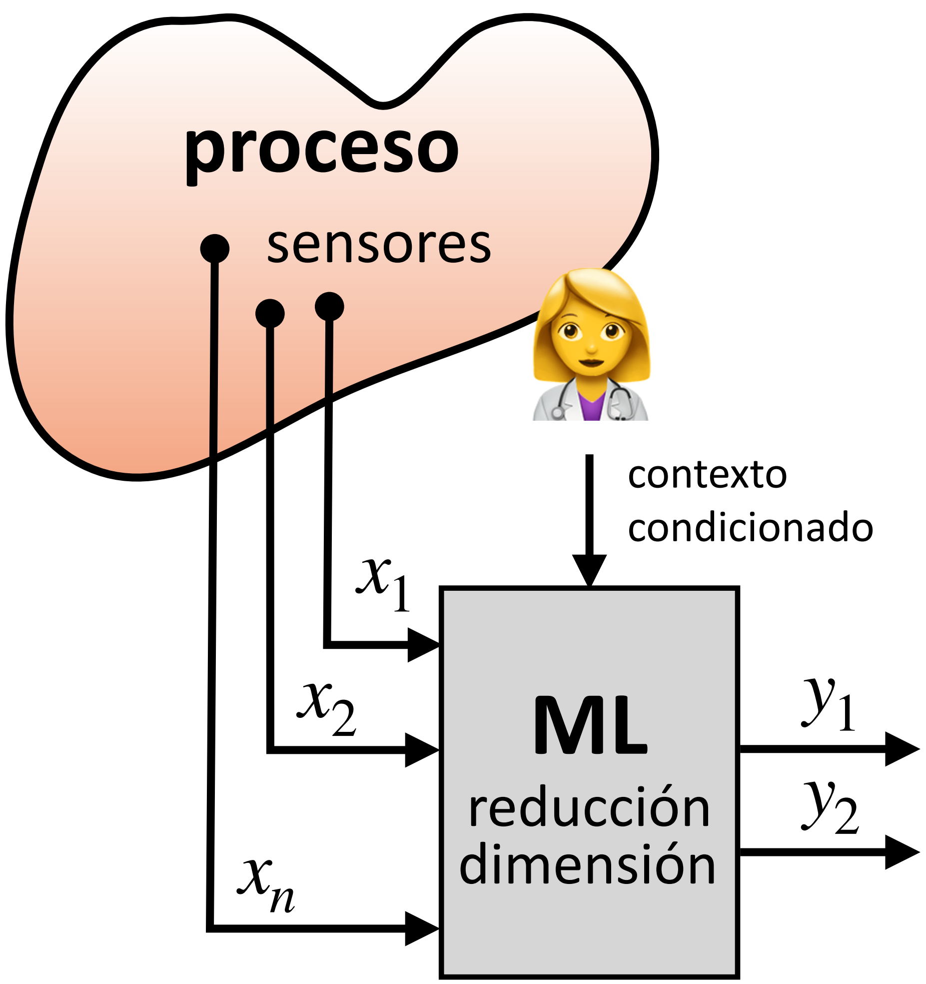
Mapas de modelos:
fuerzas de laminación



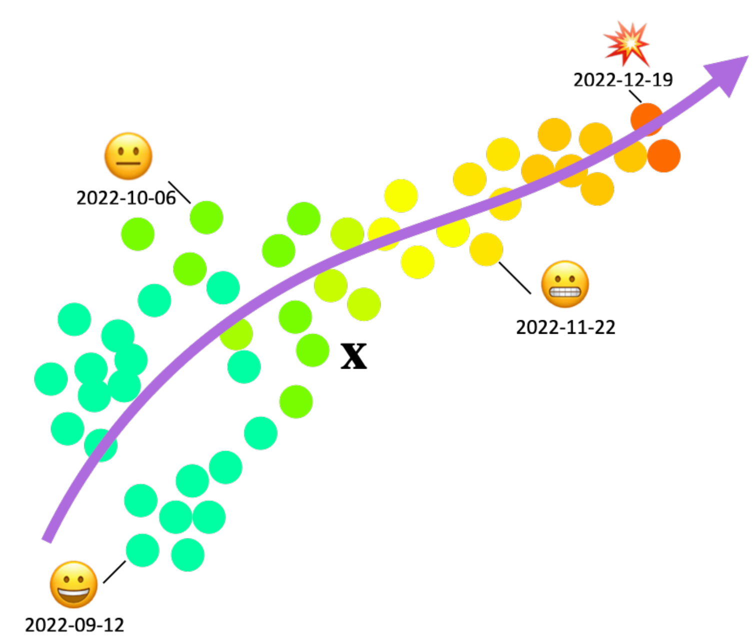
Reducción de la dimensionalidad

Visualización de datos / Extracción de características

Universidad de Oviedo

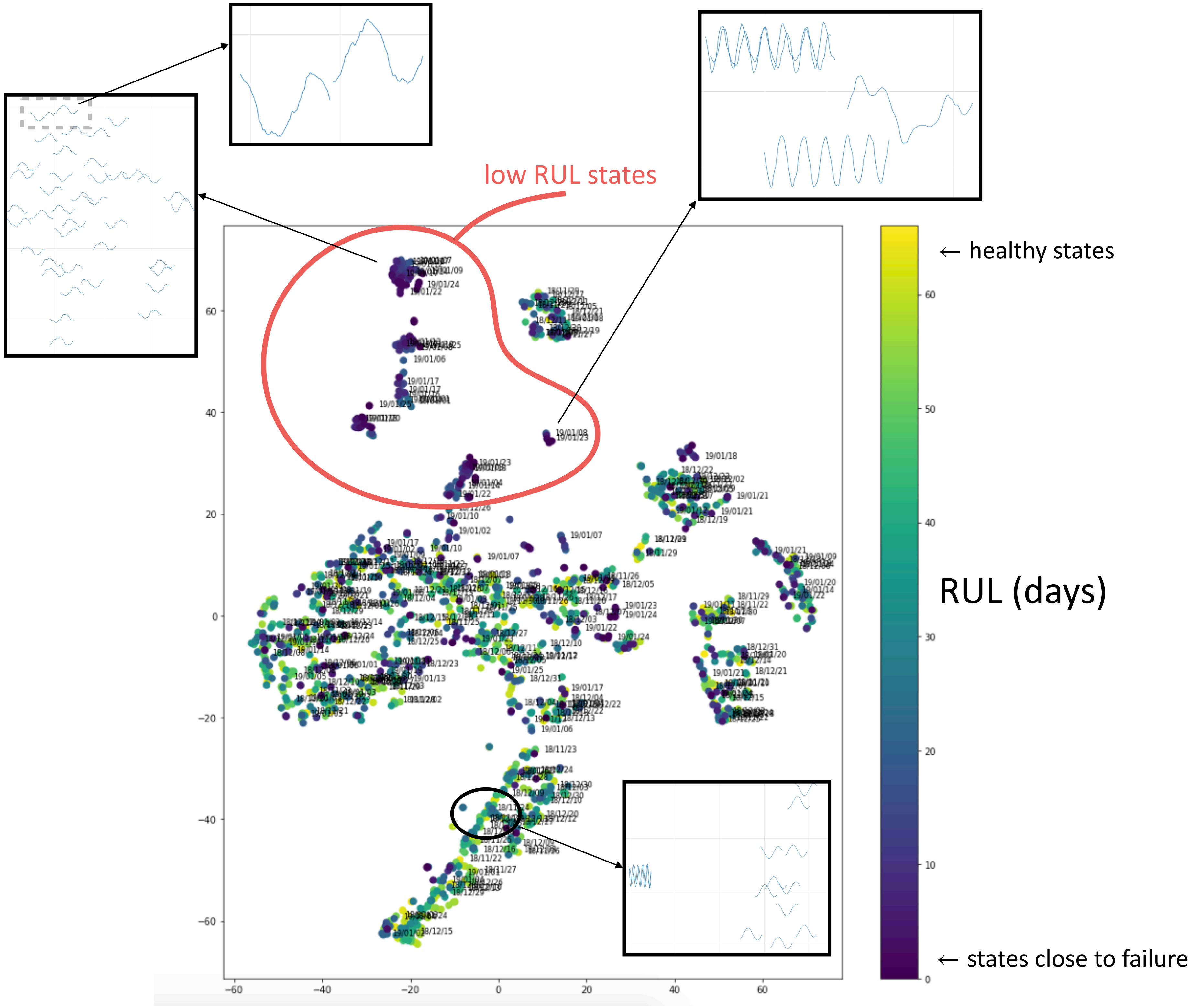


Análisis del RUL (*Remaining Useful Life)*



Reducción de la dimensionalidad

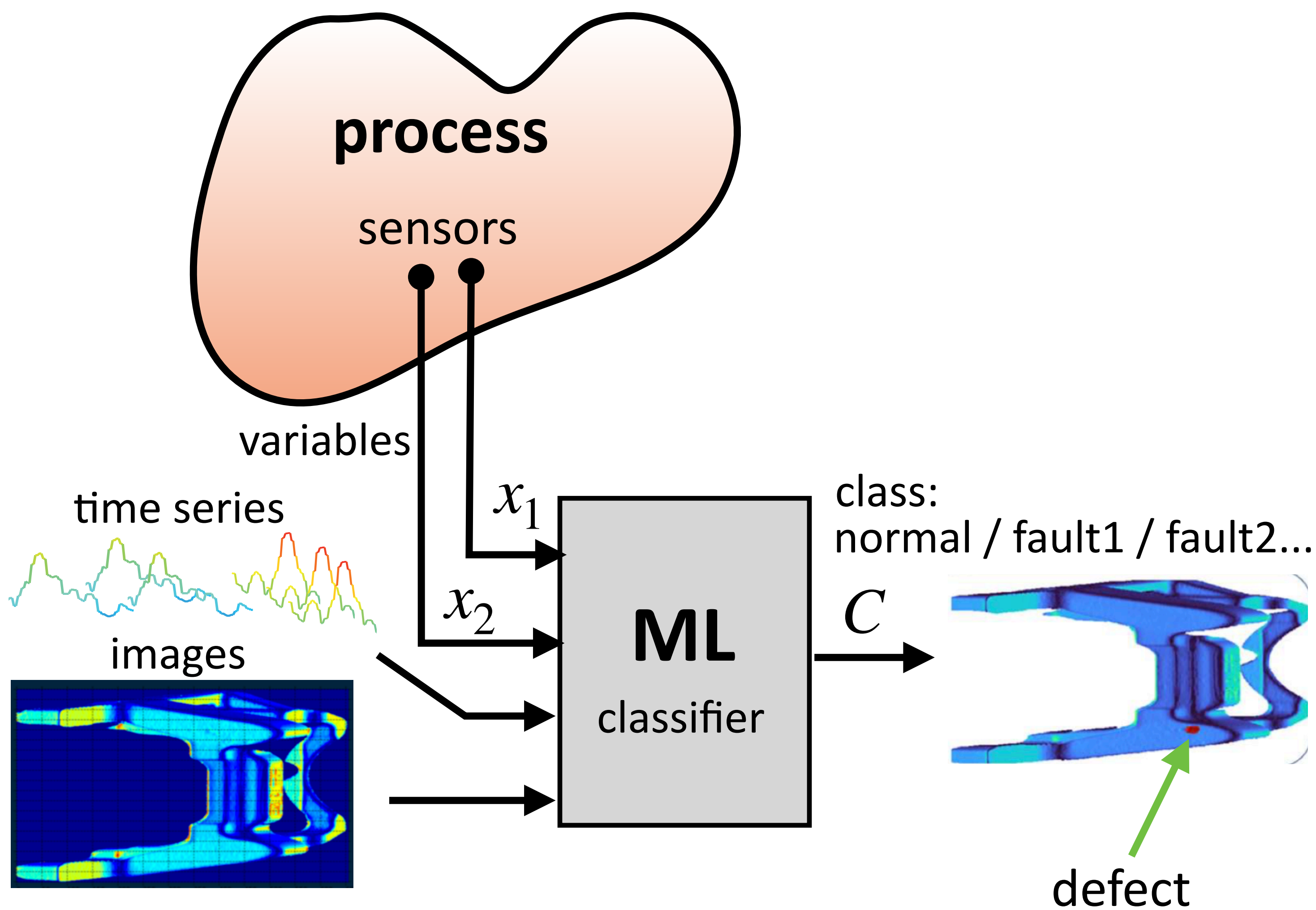
mapas de estados del proceso



work financed by ArcelorMittal
under project FUO-20-227

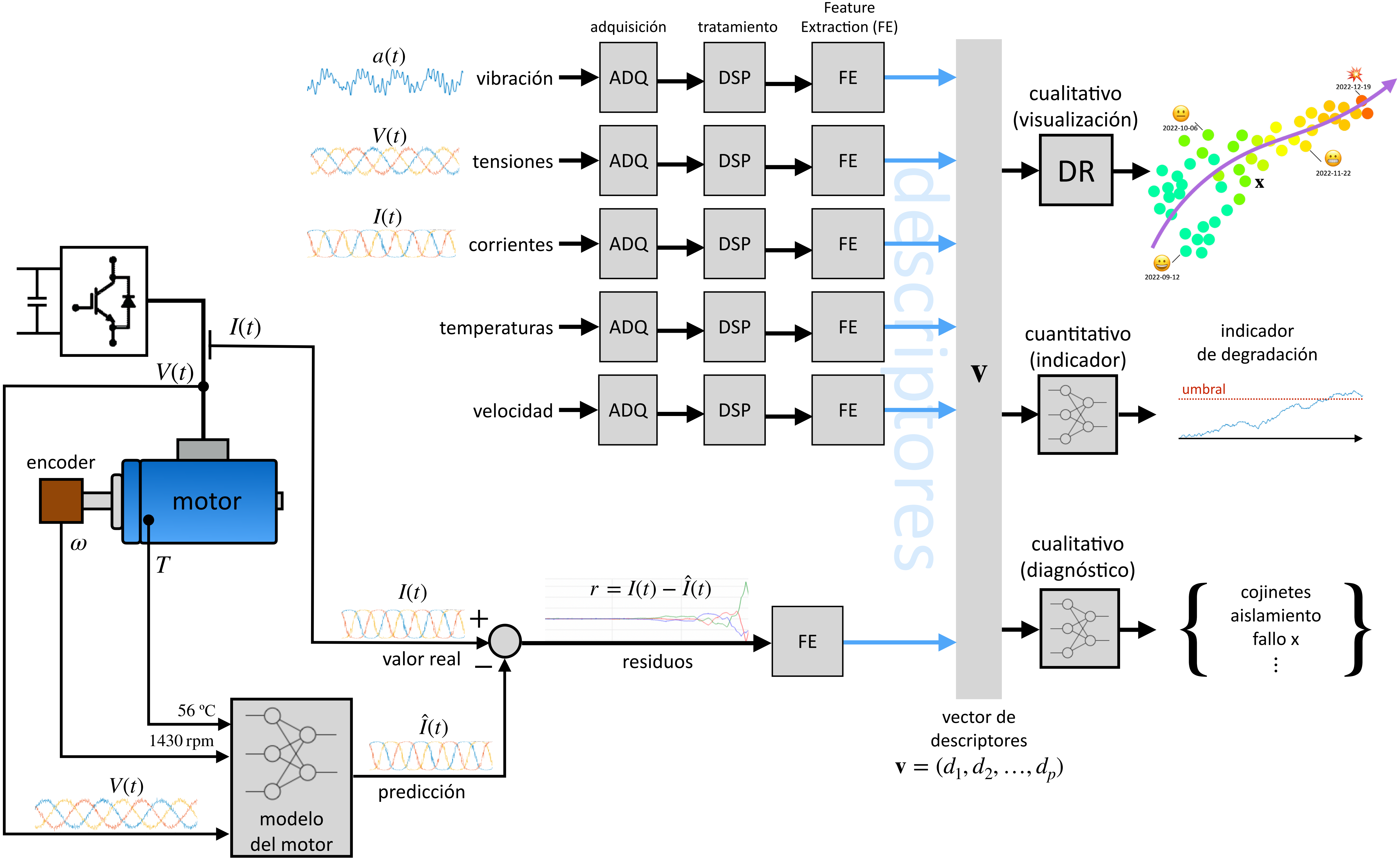
Clasificación

Diagnóstico de fallos / Monitorización de la condición



Ejemplo de aplicación

analítica de datos en motores



redundancia analítica

Conclusiones

- ML permite construir **modelos** a partir de datos
- Existen **librerías** potentes que simplifican mucho el entrenamiento e inferencia de modelos ML Python, Matlab, R, …
- Permite muchos tipos de **tareas**:
  - detección y diagnóstico de fallos
  - sensores virtuales, gemelo digital
  - predicción, pronóstico
  - análisis exploratorio / visualización de datos
- **Aplicaciones** en problemas de ingeniería eléctrica, electrónica, mecánica…
- Un **futuro** impactante
  - gemelo digital
  - modelos subrogados
  - edge / cloud computing
  - modelos de lenguage (LLM)