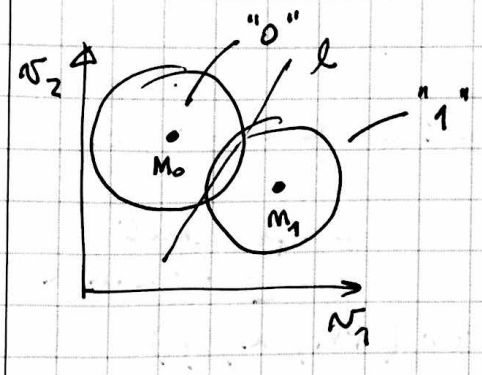
Fecha / Date:

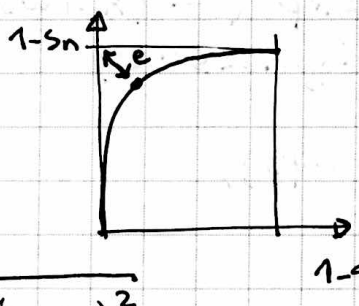
CLASIFICADOR DOS CLASES DOS CARAC.



$l: \quad a_1 v_1 + a_2 v_2 + 1 = 0$

Como encontrar (a_1, a_2)?

Ejemplo con minimización de distancia al mejor punto ROC



$$e = \sqrt{(1-S_n)^2 + (1-S_p)^2}$$

← función objetivo $e = J(a_1, a_2)$

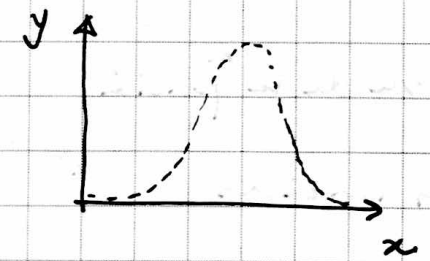
Se encuentra l tal que $e \to \text{Min}$

Implementación:

→ Se debe usar fminsearch

Ejemplo de Estimación de Parámetros No Lineales

Modelo $y = a\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$



Puntos medidos: y $\quad e = \|y - \hat{y}\| \to \text{Min}$

Notas / Notes: estimado: $\hat{y}$

↳ Vectores de N puntos

Fecha / Date:

Implementación.

Definir función que minimice el error e

→ fgausserror.m

parametros
vector x

```
[err, ys] = fgausserror(th, x, y)
```

vector y datos medidos

```
as    = th(1)
mus   = th(2)
sigmas = th(3)

ys = as * exp(-(x-mus).*(x-mu))/
     2/sigmas^2;

err = norm(y-ys);
```

→ Programa que minimiza

y ← datos medidos (ejemplo simulación de campana de Gauss con ruido

th0 ← valores iniciales de a, μ, σ

```
ths = fminsearch('fgausserror', th0, [], x, y)
```

Nombre de la función

Valores iniciales

opciones del algoritmo de minimización

Variables de la función a minimizar

Ver Carpeta Ejemplo - Estimación Parametros

Ejecutar Ejemplo Estimacion No Lineal

⟹ Como implementar fminsearch en la clasificación?

(1) Se debe tener una función que calcule el error e para dado (a_1, a_2):

→ ver función J = feroc(a, X, d)

recta ← $a = [a_1\ a_2]$

caracteristicas ← $X = [N_1\ N_2]$

clasificación ideal ← d = 1 ó 0

$w = [a_1\ a_2\ 1]^T$

XX = [X ones(n,1)] ← número de muestras en X

ds = XX * w <> 0 ← clasificación

```
TP = sum (ds .* d);
TN = sum (not(ds) .* not(d));
FN = sum ( not(ds) .* d );
FP = sum ( ds .* not(d))

Sn = TP / (TP + FN);
Sp1 = FP / (FP + TN);

J = sqrt(Sp1^2 + (1-Sn)^2)
```

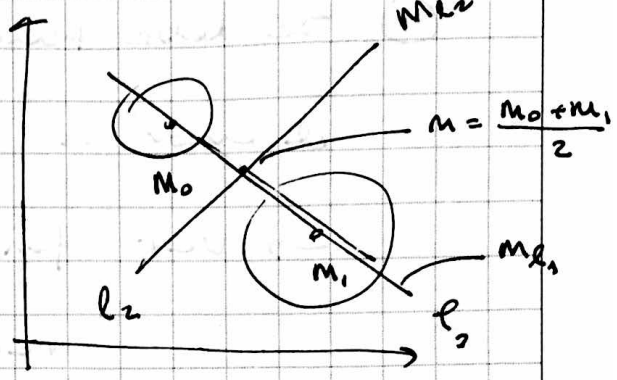
Notas / Notes:

(2) Función que estima el mejor punto de la curva

$X, d \leftarrow$ características + clasificación

Punto inicial

$a_0 \leftarrow [a_{10}\ \ a_{20}]$



$a_s \leftarrow$ fminsearch ('fcosc', a_0, [ ], X, d)

Otros criterios

(A) → Hasta ahora : $J = \sqrt{(1-s_p)^2 + (1-s_n)^2} \rightarrow \min$

(B) → Criterio de Neyman Pearson

Se minimizan los falsas alarmas sujeto a $s_n = cte.$

Recordatorio Lagrange:

Minimización $Q(x)$ sujeto a $F(x) = 0$ equivale a minimizar

$$J(x) = Q(x) + \lambda F(x)$$

↑ Factor de Lagrange

$\Rightarrow J = \lambda (s_n - cte)^2 + s_{p1}$

Notas / Notes:

(C) Asegurar TP → 100% con el mínimo número FP

$$J = \lambda (S_n - 1)^2 + S_{p1}$$

(D) Maximizar TP con FP = 0

$$J = \lambda S_{p1} + (1 - S_n)$$

(E) Minimizar el error cuadrático

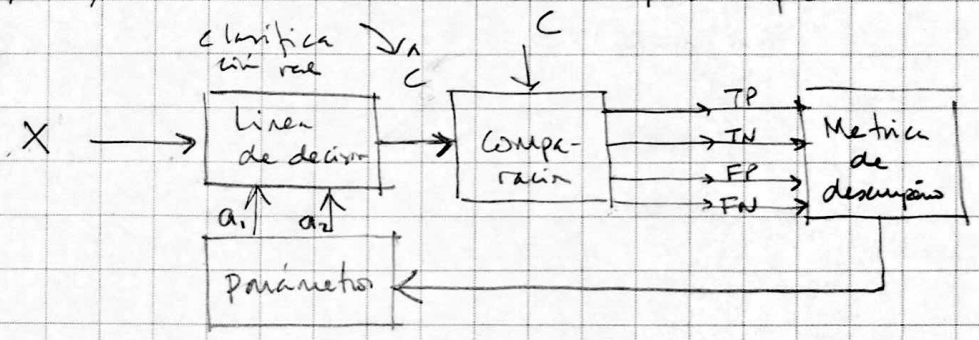
$$J = \text{norm}(d - d_s)$$

→ Fisher: $J = -\text{traza}(C_W^{-1} * C_b)$

METODOLOGÍA:

TRAINING:

(X, c) Datos de entrenamiento y clasificación ideal



Clasificador lineal: $\hat{c} = [X_1 X_2 1] \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix} > 0$

$TP = c^T \cdot \hat{c}$ $\quad S_n = \frac{TP}{TP + FN}$

$TN = \bar{c} \cdot \hat{\bar{c}}$

$FP = \bar{c} \cdot \hat{c}$ $\quad 1 - S_p = \frac{FP}{FP + TN}$

$FN = c \cdot \hat{\bar{c}}$

Hasta ahora → (A) página anterior

TESTING X_t, c_t $\quad \hat{c}_t = [X_t \; 1] \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix} > 0$

Notas / Notes:

Para un modelo cuadrático

$$XX = \begin{bmatrix} X_1^2 & X_2^2 & X_1 X_2 & X_1 & X_2 & 1 \end{bmatrix}$$

$$\underline{a} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & 1 \end{bmatrix}$$

Explicación de dmin

" de Mahalanobis

Explicación de knn.

Estructura de Bahu