Perceptron básico

Luis Rizo

UP

Acercamiento de McCulloch-Pitts , 1943

Bulletin of Mathematical Biology Vol. 52, No. 1/2, pp. 99–115, 1990.
Printed in Great Britain.

0092–8240/90$3.00 + 0.00
Pergamon Press plc
Society for Mathematical Biology

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

■ WARREN S. MCCULLOCH AND WALTER PITTS
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.

Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing circles; and that for any logical expression satisfying certain conditions, one can find a net behaving in the fashion it describes. It is shown that many particular choices among possible

Rosenblatt Perceptron´s - 1958

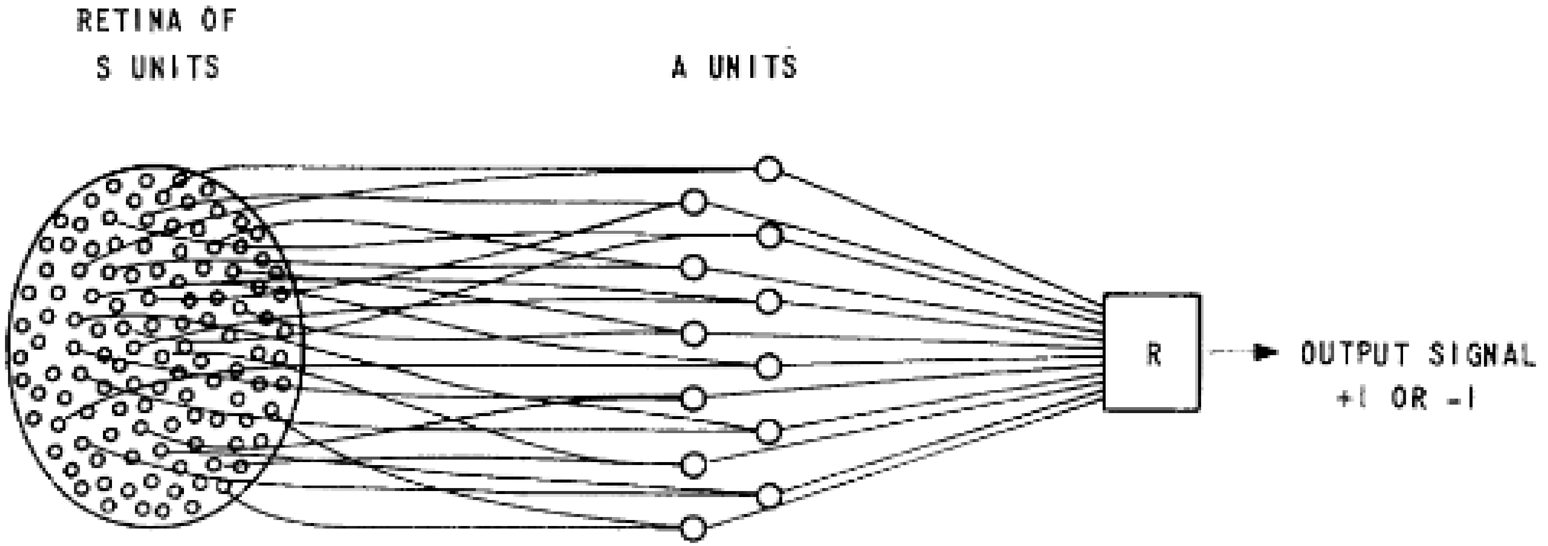


Fig. 2. Network organization of a typical elementary perceptron

SEPTEMBER 1964 *Rosenblatt—Analytic Techniques for the Study of Ne*

Rosenblatt Perceptron´s - 1958

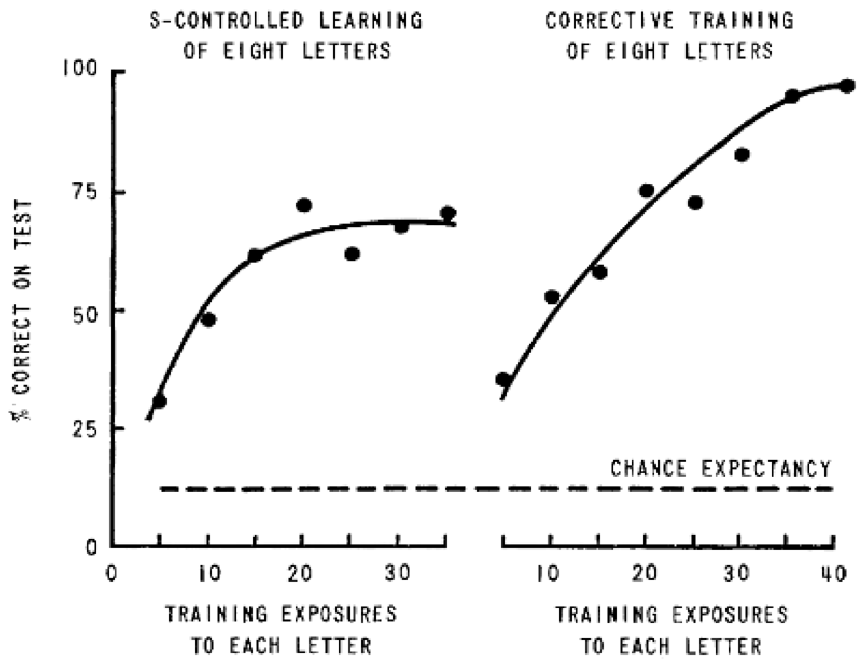


Fig. 3 (left). Learning curves for 8-letter identification task

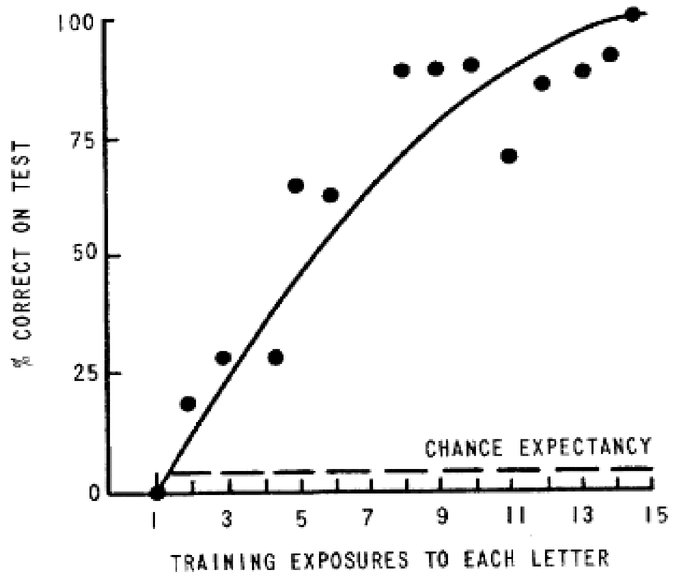
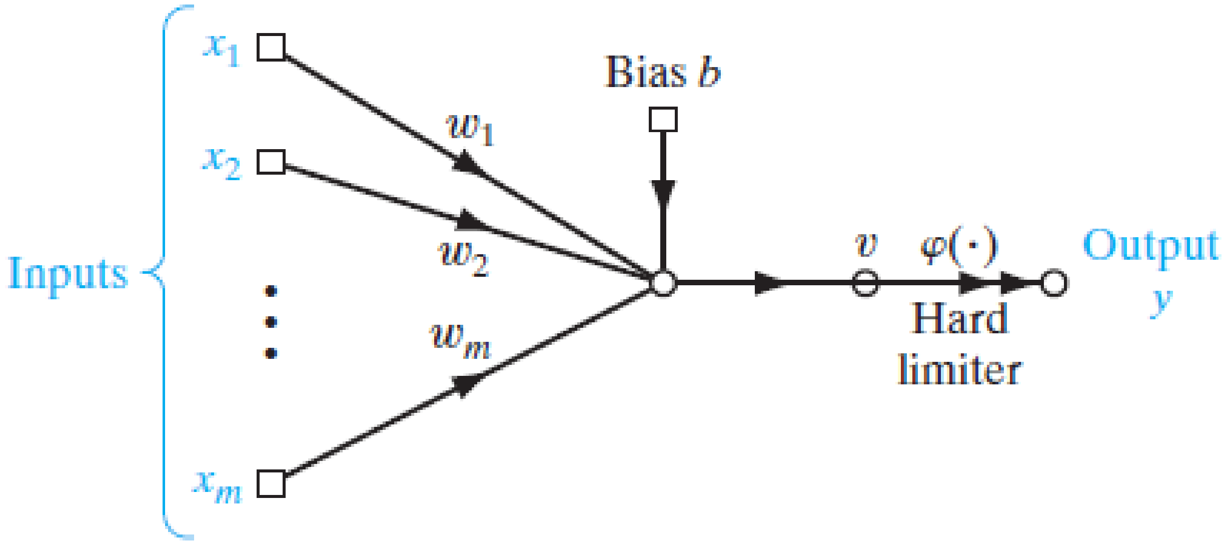


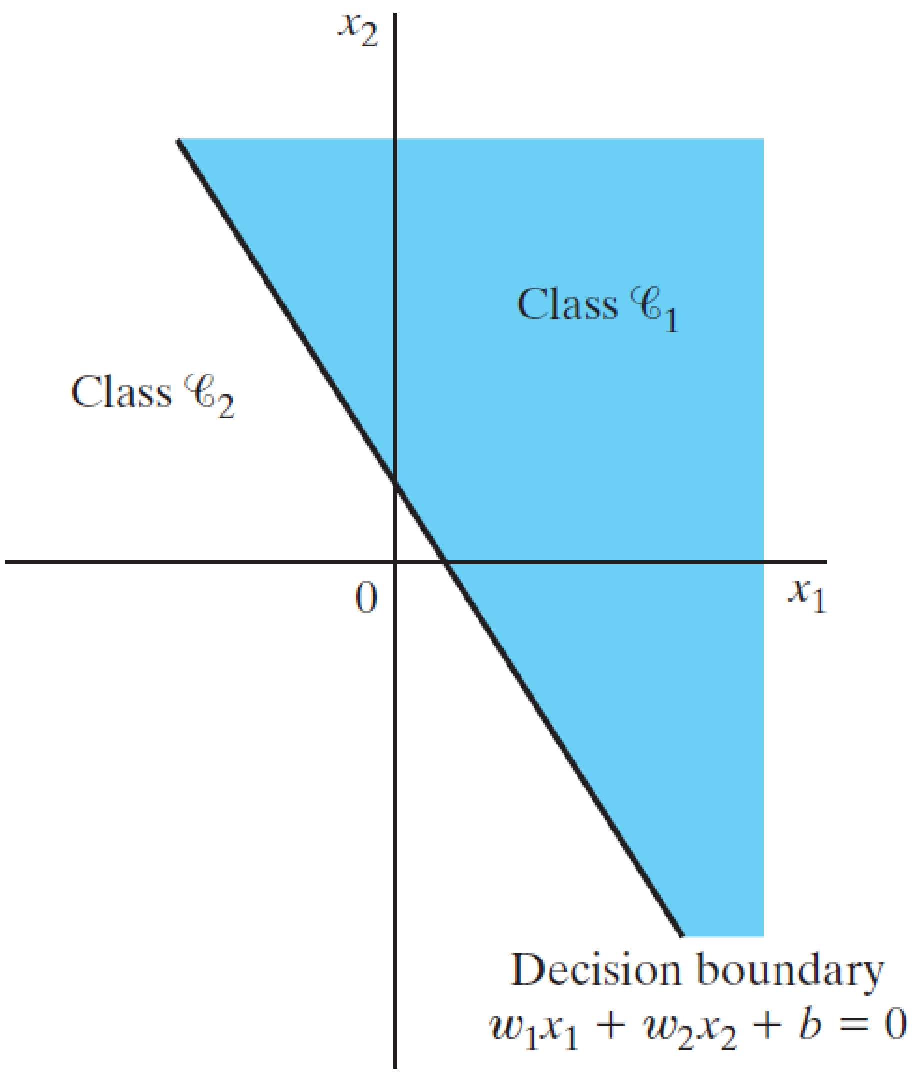
Fig. 4 (right). Learning curve for 26 letters: corrective training

Rosenblatt Perceptron´s - 1958



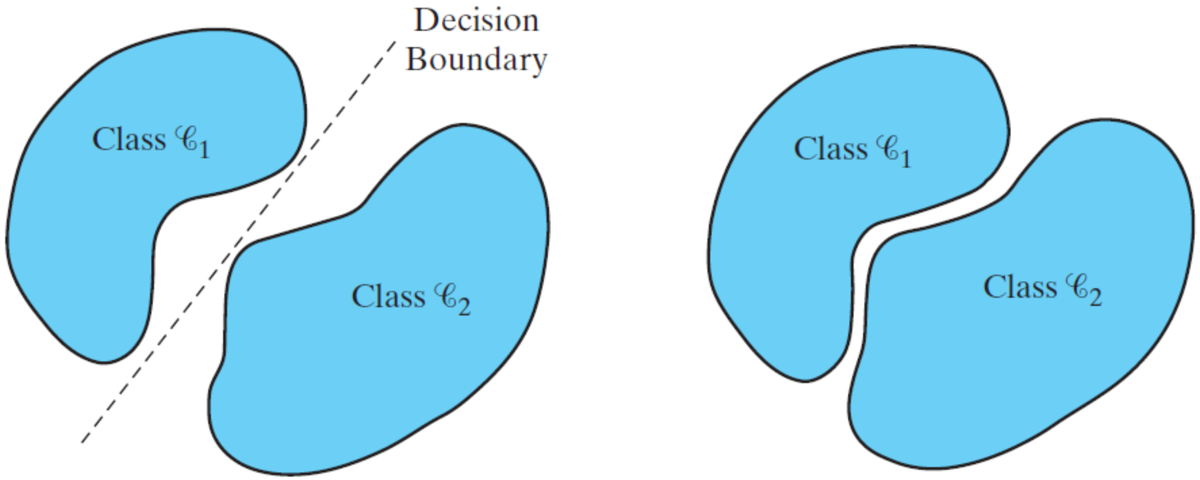
$$v = \sum_{i=1}^{m} w_i x_i + b$$

Perceptron - classification



Training perceptron

- Convergence conditions
  - The overall set of training vectors are well defined
  - The sets are separable
  - Training algorithm
    - $\mathbf{w}(n + 1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n)$ if $\mathbf{w}^T(n)\mathbf{x}(n) \;\; 0$ and $\mathbf{x}(n)$ belongs to class c1
    - $\mathbf{w}(n + 1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n)$ if $\mathbf{w}^T(n)\mathbf{x}(n) > 0$ and $\mathbf{x}(n)$ belongs to class c2



NN Approaches

Algorithmic approaching

Bayesian approaching

Training perceptron algorithm - Haykin

Variables and Parameters:

$$\begin{aligned}
\mathbf{x}(n) &= (m+1)\text{-by-1 input vector} \\
&= [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T \\
\mathbf{w}(n) &= (m+1)\text{-by-1 weight vector} \\
&= [b, w_1(n), w_2(n), \ldots, w_m(n)]^T \\
b &= \text{bias} \\
y(n) &= \text{actual response (quantized)} \\
d(n) &= \text{desired response} \\
\eta &= \text{learning-rate parameter, a positive constant less than unity}
\end{aligned}$$

1. *Initialization.* Set $\mathbf{w}(0) = \mathbf{0}$. Then perform the following computations for time-step $n = 1, 2, \ldots$.
2. *Activation.* At time-step n, activate the perceptron by applying continuous-valued input vector $\mathbf{x}(n)$ and desired response $d(n)$.
3. *Computation of Actual Response.* Compute the actual response of the perceptron as
$$y(n) = \operatorname{sgn}[\mathbf{w}^T(n)\mathbf{x}(n)]$$
where $\operatorname{sgn}(\cdot)$ is the signum function.
4. *Adaptation of Weight Vector.* Update the weight vector of the perceptron to obtain
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$
where
$$d(n) = \begin{cases} +1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_1 \\ -1 & \text{if } \mathbf{x}(n) \text{ belongs to class } \mathscr{C}_2 \end{cases}$$
5. *Continuation.* Increment time step n by one and go back to step 2.

Bayes classifier – no a priori information

- Se define la función de costo

$$J(\mathbf{w}) = \sum_{\mathbf{x}(n)\in\mathscr{X}} (-\mathbf{w}^T\mathbf{x}(n)d(n))$$

- Se minimiza buscando el sentido contrario del gradiente

$$\nabla J(\mathbf{w}) = \sum_{\mathbf{x}(n)\in\mathscr{X}} (-\mathbf{x}(n)d(n)) \qquad \boldsymbol{\nabla} = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \dots, \frac{\partial}{\partial w_m}\right]^T$$

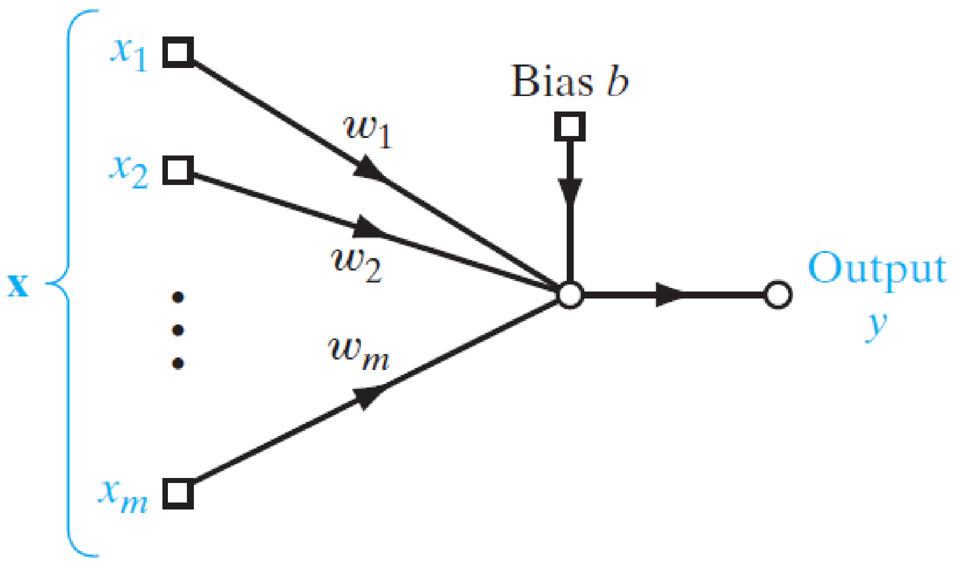
- Quedando el entrenamiento de la red como:

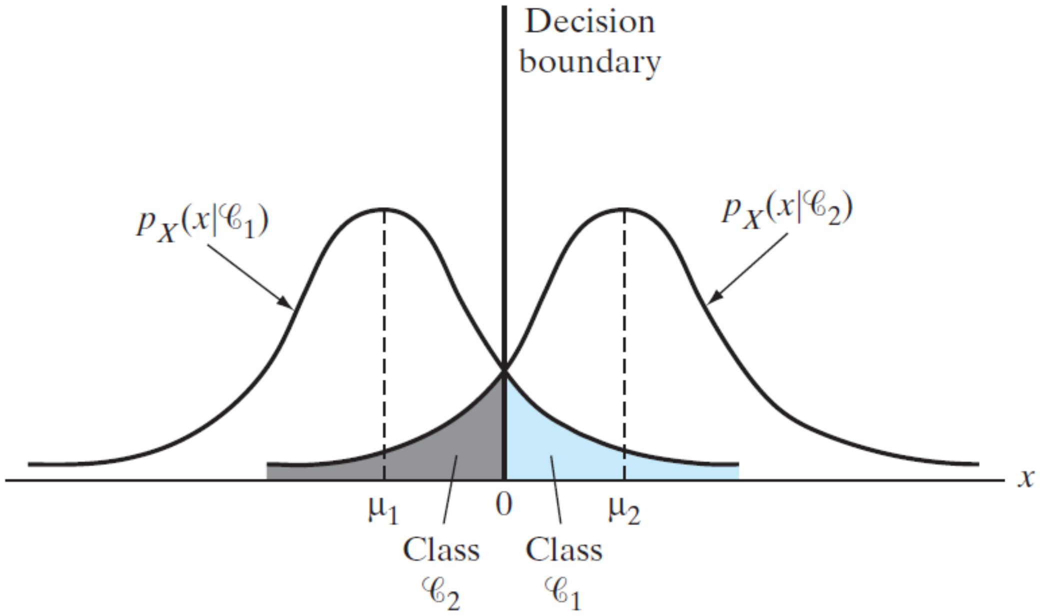
$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) - \eta(n)\nabla J(\mathbf{w}) \\ &= \mathbf{w}(n) + \eta(n)\sum_{\mathbf{x}(n)\in\mathscr{X}} \mathbf{x}(n)d(n)\end{aligned}$$

Actividad

- Ingresar al sitio:
  - https://playground.tensorflow.org/
  - Configurar la red de la siguiente manera:
    - Learning rate 0.03
    - Activation Linear
    - Regularization: none
    - Problem Type: classification
    - 0 Hidden layers
  - Variar Learning rate y observar test loss and training loss
  - Variar las formas a clasificar

Gaussian Classifier

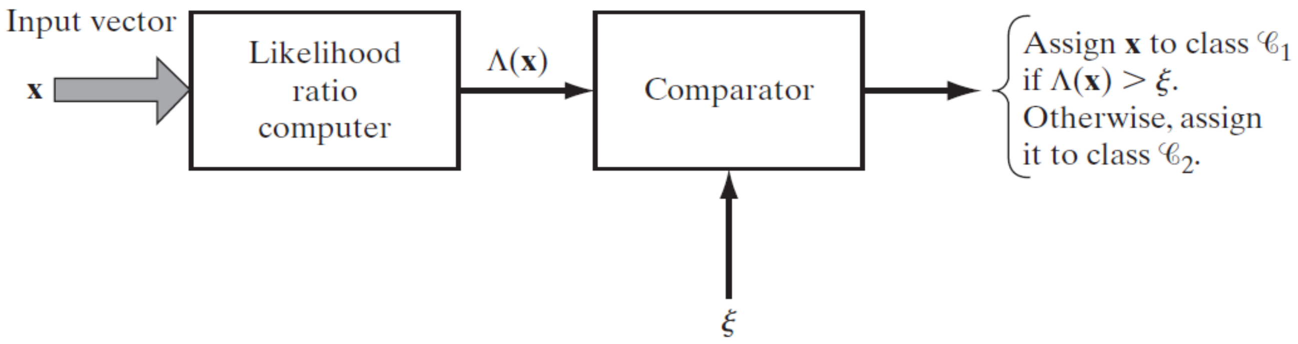




Bayes classifier – minimun average risk

$$\mathcal{R} = c_{11}p_1 \int_{\mathcal{H}_1} p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{22}p_2 \int_{\mathcal{H}_2} p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x}$$
$$+ c_{21}p_1 \int_{\mathcal{H}_2} p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)d\mathbf{x} + c_{12}p_2 \int_{\mathcal{H}_1} p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)d\mathbf{x}$$

$$\Lambda(\mathbf{x}) = \frac{p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_1)}{p_{\mathbf{X}}(\mathbf{x}|\mathcal{C}_2)}$$



Bayes classifier – minimun average risk

- Función de densidad de probabilidad para el caso Gaussiano

$$p_{\mathbf{X}}(\mathbf{x}|\mathscr{C}_i) = \frac{1}{(2\pi)^{m/2}(\det(\mathbf{C}))^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right), \quad i = 1, 2$$

- Aplicando el logaritmo de la función de verosimilitud

$$\begin{aligned} \log\Lambda(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T\mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T\mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T\mathbf{C}^{-1}\mathbf{x} + \frac{1}{2}(\boldsymbol{\mu}_2^T\mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\mathbf{C}^{-1}\boldsymbol{\mu}_1) \end{aligned}$$

- Quedando la neurona con los siguientes pesos (problema lineal)

$$y = \mathbf{w}^T\mathbf{x} + b$$

$$\begin{aligned} \mathbf{y} &= \log\Lambda(\mathbf{x}) \\ \mathbf{w} &= \mathbf{C}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ b &= \frac{1}{2}(\boldsymbol{\mu}_2^T\mathbf{C}^{-1}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^T\mathbf{C}^{-1}\boldsymbol{\mu}_1) \end{aligned}$$

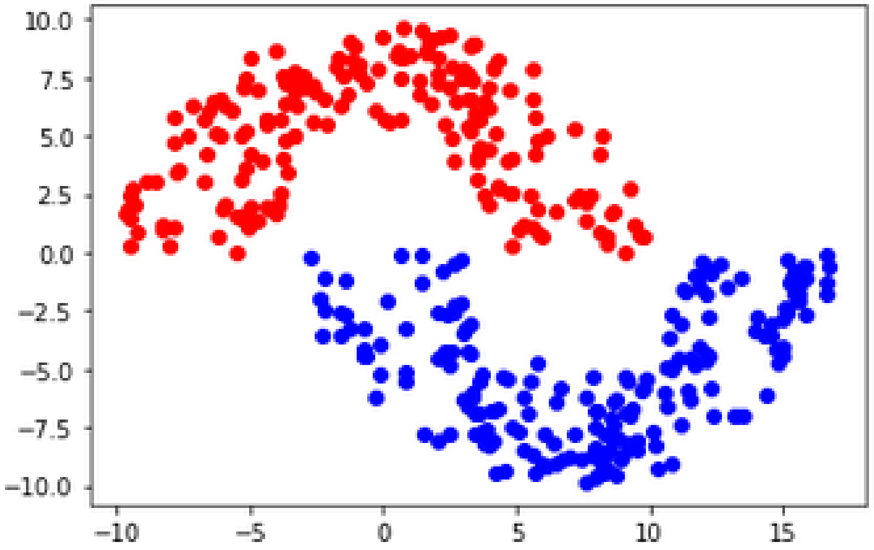
Actividad

- Generar un set de datos
  - 70% entrenamiento
  - 30% pruebas
- Clasificarlo para diferentes distancias entre el conjunto azul y el conjunto rojo
- Se puede utilizar:
  - Algoritmo clásico del Perceptrón
  - Bayes con gradiente descendiente

Testing data

```python
from random import random
import matplotlib.pyplot as plt
import math

points = 200

x1 = [0 for _ in range(points)]
y1 = [0 for _ in range(points)]
x2 = [0 for _ in range(points)]
y2 = [0 for _ in range(points)]

for i in range(points):
    d = 0
    r = 4
    w = 6
    # 0 ~ 180
    a = random()*math.pi
    x1[i] = math.sqrt(random()) * math.cos(a)*(w/2) + ((-(r+w/2) if(random() < 0.5) else (r+w/2)) * math.cos(a))
    y1[i] = math.sqrt(random()) * math.sin(a)*(w) + (r * math.sin(a)) - d
    # 180 ~ 360
    a = random()*math.pi + math.pi
    x2[i] = (r+w/2) + math.sqrt(random()) * math.cos(a)*(w/2) + ((-(r+w/2)) if(random() < 0.5) else (r+w/2)) * math.cos(a)
    y2[i] = -(math.sqrt(random()) * math.sin(a)*(-w) + (-r * math.sin(a))) + d

plt.scatter(x1, y1, color="r")
plt.scatter(x2, y2, color="b")
plt.show()
```

```python
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X, y);
```



Tarea

Set de datos de problema

- Clasificación/regresión

Problema práctico a abordar

Hacer equipos

Remarks

- Definición del problema (clasificación o regresión)
- Definición de la función de costo
- Resolución de la función de costo para el entrenamiento:
  - Gradiente
  - Newton
  - ….
- Entrenamiento
- Preparación de los datos
  - Técnica de Hold Out (70% entrenamiento, 30% prueba)
- Validación (producción)

Ir más allá

- ¿Mejor clasificador para la rueda, Bayes-Neural o Regresión Logística?
- ¿Teorema del límite central?
- Tema nuevos, evaluar las proabilidades sin conocer las funciones de densidad mediante:
  - ¿Desigualdad de Cauchy-Schwarz?
  - ¿Hoeffding's inequality? (Kullback Distance Bound)