Neuronas artificiales

MANADINE – Análisis de Datos 1

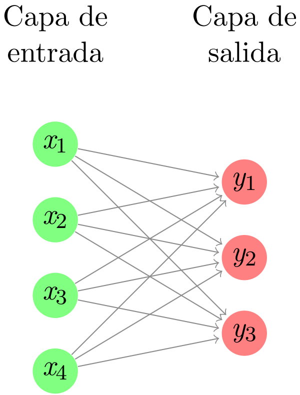
2 de mayo de 2023

Introducción

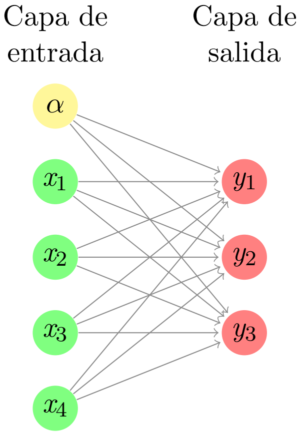
- inspirado en las conexiones (sinapsis) entre neuronas cerebrales
- permiten clasificación o regresión
- son modelos de regresión no lineal con muchos parámetros (caja negra: el ajuste no es constructivo)
- permiten ajustar información no estructurada (fotos...) pero `https://techcrunch.com/2018/01/02/these-psychedelic-stickers-blow-ai-minds/`
- tipos
  - perceptrón
  - base radial
  - mapas autoorganizados
  - ...

Índice

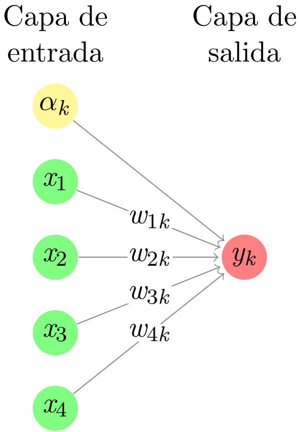
Perceptrón simple



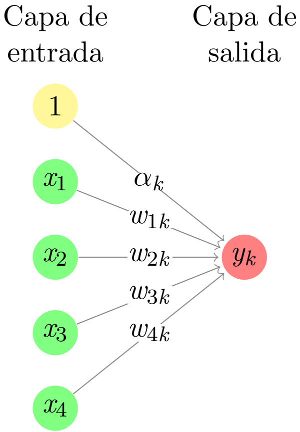
Perceptrón simple



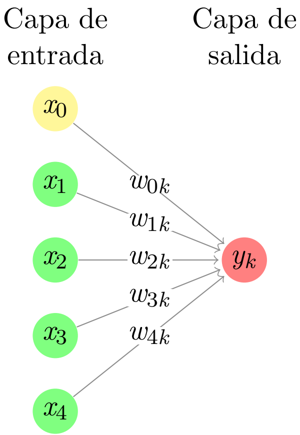
Perceptrón simple



Perceptrón simple



Perceptrón simple



Perceptrón simple

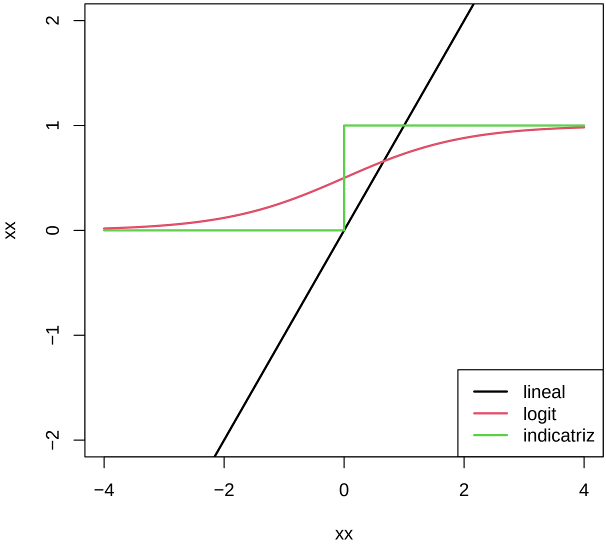
- $\mathscr{i}$ = entrada (*input*)
- $\mathscr{o}$ = salida (*output*)
- ϕ = función de activación
- α = constante, sesgo (*bias*)
- w = peso (*weight*), coeficiente de sinapsis

$$y_k = \phi_{\mathscr{o}} \left(\alpha_k + \sum_{i=1}^{I} w_{ik}\, x_i \right) = \phi_{\mathscr{o}} \left(\sum_{i=0}^{I} w_{ik}\, x_i \right)$$

Perceptrón simple

- funciones de activación ϕ habituales
  - lineal
$$\phi_o(x) = x$$
  - logística
$$\phi_o(x) = \ell(x) = \frac{\exp(x)}{1 + \exp(x)}$$
  - indicatriz, umbral, característica
$$\phi_o(x) = \mathbb{1}_{[0,\infty)}(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geqslant 0 \end{cases}$$

Perceptrón simple



Perceptrón simple: ajuste

- función de activación:
  - para respuesta dicótoma, logística: $\phi_o(x) = \frac{\exp(x)}{1+\exp(x)}$
  - para tres o más categorías, lineal: $\phi_o(x) = x$
- criterios de ajuste
  $p =$ patrón $\quad t =$ respuesta $\in \{0; 1\}$ $\quad y =$ predicción
  - para respuesta dicótoma, entropía $(0 \leqslant y \leqslant 1)$

$$E = \sum_p \sum_k \left[t_k^{(p)} \ln \frac{t_k^{(p)}}{y_k^{(p)}} + (1 - t_k^{(p)}) \ln \frac{1 - t_k^{(p)}}{1 - y_k^{(p)}} \right]$$

  - para respuesta múltiple, *softmax* $(y \in \mathbb{R})$

$$E = \sum_p \sum_k -t_k^{(p)} \log \widehat{\Pr}[p \in k] \qquad \widehat{\Pr}[p \in k] = \frac{\exp\left(y_k^{(p)}\right)}{\sum_{c=1}^K \exp\left(y_c^{(p)}\right)}$$

Perceptrón simple

```
> ## para ahorrar espacio en esta presentación:
> options (width = 58)
> names(iris)[1:4] <- c("Lsep","Asep","Lpet","Apet")
> library (nnet)   # biblioteca distribuida con R básico
> red <- nnet (Species ~ ., iris, size = 0, skip = TRUE)
# weights:  15
initial  value 285.325363
iter  10 value 8.096373
iter  20 value 5.960549
iter  30 value 5.952579
iter  40 value 5.949318
iter  50 value 5.949290
final  value 5.949277
converged
```

Perceptrón simple

```
> red
a 4-0-3 network with 15 weights
inputs: Lsep Asep Lpet Apet
output(s): Species
options were - skip-layer connections  softmax modelling
```

Perceptrón simple

```
> summary (red)
a 4-0-3 network with 15 weights
options were - skip-layer connections  softmax modelling
 b->o1 i1->o1 i2->o1 i3->o1 i4->o1
 -2.82   1.96  17.93 -14.07 -11.96
 b->o2 i1->o2 i2->o2 i3->o2 i4->o2
 22.60   0.21  -5.82   2.29  -2.75
 b->o3 i1->o3 i2->o3 i3->o3 i4->o3
-20.08  -2.26 -12.51  11.73  15.55
```

Perceptrón simple

```
> names (red)
 [1] "n"             "nunits"       "nconn"
 [4] "conn"          "nsunits"      "decay"
 [7] "entropy"       "softmax"      "censored"
[10] "value"         "wts"          "convergence"
[13] "fitted.values" "residuals"    "lev"
[16] "call"          "terms"        "coefnames"
[19] "xlevels"
```

Perceptrón simple

```
> red $ wts
 [1]  -2.8239735   1.9633937  17.9281992 -14.0713310
 [5] -11.9612859  22.6031112   0.2066642  -5.8225870
 [9]   2.2915637  -2.7476730 -20.0809056  -2.2580146
[13] -12.5066021  11.7273984  15.5503582
> head (red $ fitted.values)
  setosa   versicolor    virginica
1      1 6.295937e-19 9.314251e-46
2      1 1.285481e-13 8.803349e-39
3      1 3.076484e-16 3.526606e-42
4      1 1.040148e-13 1.964745e-38
5      1 6.980244e-20 6.771966e-47
6      1 2.379159e-20 7.638111e-46
```

Perceptrón simple

```
> head (red $ fitted.values)
  setosa   versicolor    virginica
1      1 6.295937e-19 9.314251e-46
2      1 1.285481e-13 8.803349e-39
3      1 3.076484e-16 3.526606e-42
4      1 1.040148e-13 1.964745e-38
5      1 6.980244e-20 6.771966e-47
6      1 2.379159e-20 7.638111e-46
> summary (apply (red $ fitted.values, 1, sum))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       1       1       1       1       1
```

Perceptrón simple

```
> red $ value
[1] 5.949277
> indices.fila     <- 1 : nrow (iris)
> indices.columna <- match (iris$Species,
+                             levels(iris$Species))
> indices <- cbind (indices.fila, indices.columna)
> - sum (log (red$fitted.values [indices]))
[1] 5.949277
```

Perceptrón simple

```
> aggregate (iris[,1:4], list(iris$Species), median)
     Group.1 Lsep Asep Lpet Apet
1     setosa  5.0  3.4 1.50  0.2
2 versicolor  5.9  2.8 4.35  1.3
3  virginica  6.5  3.0 5.55  2.0
> flor <- data.frame(Lsep=6,Asep=2.9,Lpet=5,Apet=1.7)
> predict (red, flor)
        setosa versicolor virginica
1 2.509364e-20  0.1930841 0.8069159
> predict (red, flor, type="class")
[1] "virginica"
```

Perceptrón simple

```
> predict (red, flor)
        setosa versicolor virginica
1 2.509364e-20  0.1930841 0.8069159
> summary (red)
a 4-0-3 network with 15 weights
options were - skip-layer connections  softmax modelling
 b->o1 i1->o1 i2->o1 i3->o1 i4->o1
 -2.82   1.96  17.93 -14.07 -11.96
 b->o2 i1->o2 i2->o2 i3->o2 i4->o2
 22.60   0.21  -5.82   2.29  -2.75
 b->o3 i1->o3 i2->o3 i3->o3 i4->o3
-20.08  -2.26 -12.51  11.73  15.55
```

Perceptrón simple

```
> predict (red, flor)
        setosa versicolor virginica
1 2.509364e-20  0.1930841 0.8069159
> flor1 <- c (1, as.numeric (flor))
> e1 <- exp (as.numeric (flor1 %*% red$wts[1:5]))
> e2 <- exp (as.numeric (flor1 %*% red$wts[6:10]))
> e3 <- exp (as.numeric (flor1 %*% red$wts[11:15]))
> c(e1,e2,e3) / (e1+e2+e3)
[1] 2.509364e-20 1.930841e-01 8.069159e-01
```

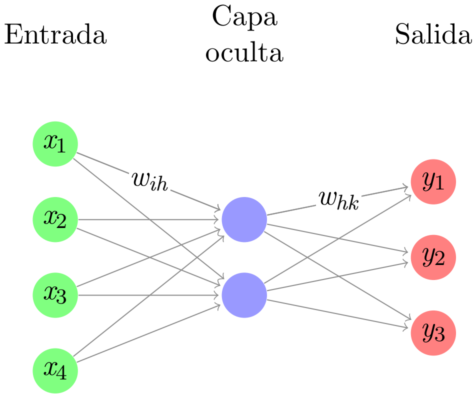
Respuesta dicótoma

```
> red2 <- nnet (factor(am)~mpg, mtcars,
+                  size=0, skip=TRUE, trace=FALSE)
> predict (red2, data.frame (mpg = 20))
       [,1]
1 0.3862832
> 1 / (1 + 1/exp (red2$wts[1] + red2$wts[2] * 20)) #logit
[1] 0.3862832
> red2 $ value
[1] 14.83758
> p <- red2 $ fitted.values                #una sola columna
> - sum (p * log(p) + (1-p) * log(1-p)) #entropía
[1] 14.83758
```

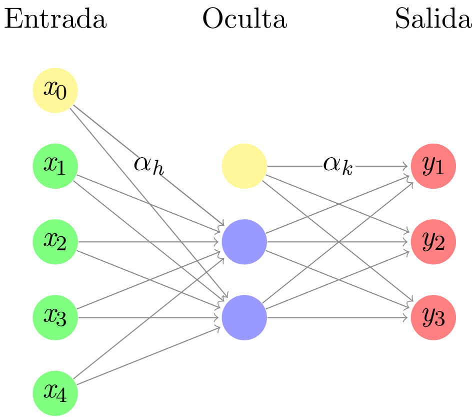
Respuesta dicótoma

```
> p <- red2 $ fitted.values                    #una sola columna
> - sum (p * log(p) + (1-p) * log(1-p)) #entropía
[1] 14.83758
> ## equivale a
> t <- +(mtcars$am == mtcars$am[1])
> n0 <- function (x) ifelse (is.na(x), 0, x)
> sum (n0(t*log(t/p)) + n0((1-t)*log((1-t)/(1-p))))
[1] 14.83758
```

Perceptrón multicapa (1 oculta)



Perceptrón multicapa (1 oculta)



Perceptrón multicapa (1 oculta)

- h = índice de neuronas en capa oculta (*hidden*)

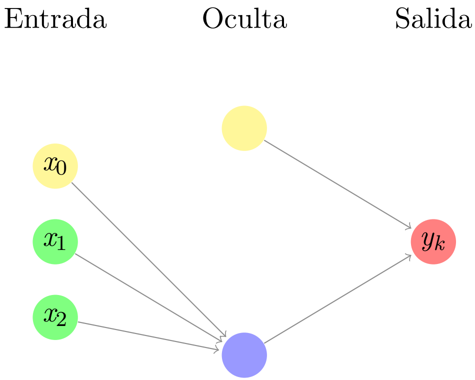
$$y_k = \phi_o\left(\alpha_k + \sum_{h=1}^{H} w_{hk}\,\phi_h\left(\alpha_h + \sum_{i=1}^{I} w_{ih}\,x_i\right)\right)$$

$$y_k = \phi_o\left(\sum_{h=0}^{H} w_{hk}\,\phi_h\left(\sum_{i=0}^{I} w_{ih}\,x_i\right)\right)$$

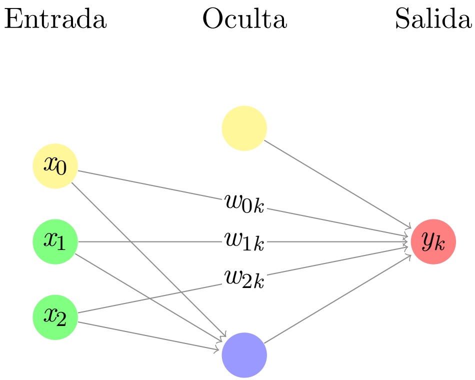
- ϕ casi siempre logística en la oculta

$$\phi_h(x) = \ell(x) = \frac{\exp(x)}{1+\exp(x)}$$

Perceptrón multicapa (1 oculta, `skip=FALSE`)



Perceptrón multicapa (1 oculta, `skip=TRUE`)



Perceptrón multicapa (1 oculta)

- `skip=FALSE`

$$y_k = \phi_o \left(\sum_h w_{hk} \, \phi_h \left(\sum_i w_{ih} \, x_i \right) \right)$$

- `skip=TRUE`

$$y_k = \phi_o \left(\sum_h w_{hk} \, \phi_h \left(\sum_i w_{ih} \, x_i \right) + \sum_i w_{ik} \, x_i \right)$$

- las conexiones *skip* pueden facilitar la interpretación de la red neuronal

Perceptrón multicapa (1 oculta)

```
> red0 <- nnet (Species ~ ., iris, size = 2,
+                  skip = FALSE, trace = FALSE)
> red0
a 4-2-3 network with 19 weights
inputs: Lsep Asep Lpet Apet
output(s): Species
options were - softmax modelling
> red1 <- nnet (Species ~ ., iris, size = 2,
+                  skip = TRUE, trace = FALSE)
> red1
a 4-2-3 network with 31 weights
inputs: Lsep Asep Lpet Apet
output(s): Species
options were - skip-layer connections  softmax modelling
```

Perceptrón multicapa (1 oculta)

```
> summary (red0)
a 4-2-3 network with 19 weights
options were - softmax modelling
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1
   1.03    0.13    0.34   -0.50   -0.92
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2
  -1.62   -7.56   -3.33   -5.56   -1.92
  b->o1  h1->o1  h2->o1
 -48.26  104.01   -1.35
  b->o2  h1->o2  h2->o2
  11.49    1.00    0.28
  b->o3  h1->o3  h2->o3
  37.11 -105.44    0.74
```

Perceptrón multicapa (1 oculta)

```
> summary (red1)
a 4-2-3 network with 31 weights
options were - skip-layer connections  softmax modelling
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1
 -4.93 -29.72 -17.20 -14.72  -4.16
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2
  1.31   0.32   4.72  -8.49  -4.16
 b->o1 h1->o1 h2->o1 i1->o1 i2->o1 i3->o1 i4->o1
  3.02  -5.74   8.73   3.14   4.90  -8.63  -8.23
 b->o2 h1->o2 h2->o2 i1->o2 i2->o2 i3->o2 i4->o2
 20.05  -4.76 -21.75  -0.42   0.69  -1.00  -4.73
 b->o3 h1->o3 h2->o3 i1->o3 i2->o3 i3->o3 i4->o3
-22.57   9.81  13.76  -2.89  -5.99   8.43  13.55
```

Regresión

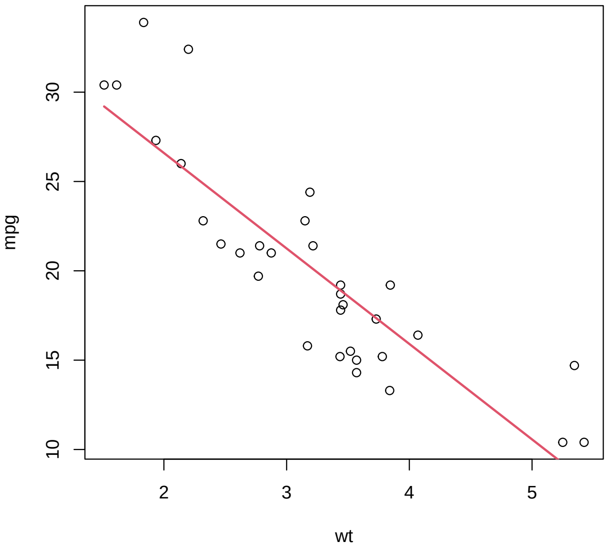
- función de salida lineal: $\phi_o(x) = x$
- teorema
  - sea f cualquier función continua sobre un compacto
  - se puede aproximar f uniformemente
  - basta incrementar el número de neuronas en la capa oculta
- la aproximación es "no constructiva"
- criterios de ajuste (p=patrón, t=objetivo, y=predicción)
  - mínimos cuadrados: $E = \sum_p \|t^{(p)} - y^{(p)}\|^2$

Regresión

```
> reg <- lm (mpg ~ wt, mtcars)
> print (summary (reg))
[...]

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16
wt           -5.3445     0.5591  -9.559 1.29e-10

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared: 0.7528, Adjusted R-squared: 0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

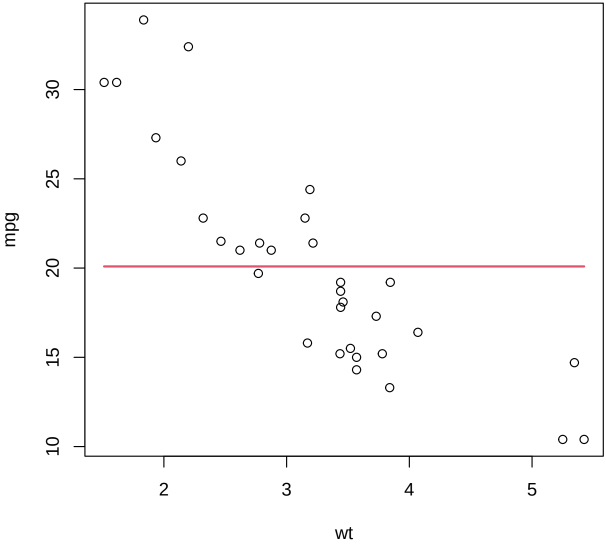

mpg
wt
10
15
20
25
30
2
3
4
5

Regresión

```
> red <- nnet (mpg ~ wt, mtcars, size = 2,
+                    linout = TRUE)
# weights:  7
initial  value 12810.488754
iter  10 value 1126.058572
final  value 1126.047409
converged
> summary (red)
a 1-2-1 network with 7 weights
options were - linear output units
 b->h1 i1->h1
  7.78  10.47
 b->h2 i1->h2
  4.49   6.21
 b->o h1->o h2->o
 3.52  9.04  7.53
```

Regresión

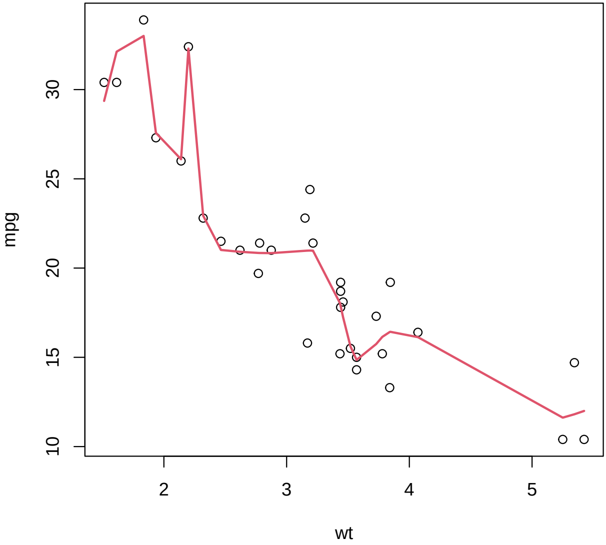
```
> red $ value
[1] 1126.047
> sum (red $ residuals ^ 2)
[1] 1126.047
> sum (reg $ residuals ^ 2)
[1] 278.3219
```



mpg
30
25
20
15
10
2
3
4
5
wt

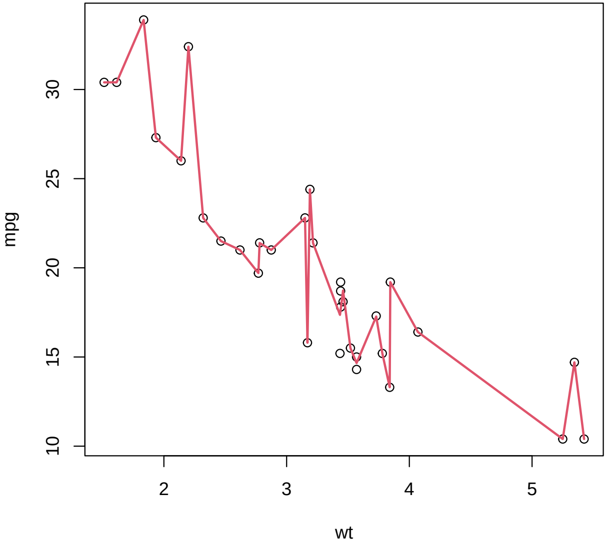
Regresión

```
> red <- nnet (mpg ~ wt, mtcars, size = 100,
+              linout = TRUE)
# weights:  301
initial  value 15327.887085
iter  10 value 212.152796
iter  20 value 204.778990
iter  30 value 197.535929
iter  40 value 177.289105
iter  50 value 167.455132
iter  60 value 150.114839
iter  70 value 142.754910
iter  80 value 123.043295
iter  90 value 99.143923
iter 100 value 93.573309
final  value 93.573309
stopped after 100 iterations
```


mpg
wt
10
15
20
25
30
2
3
4
5

Regresión

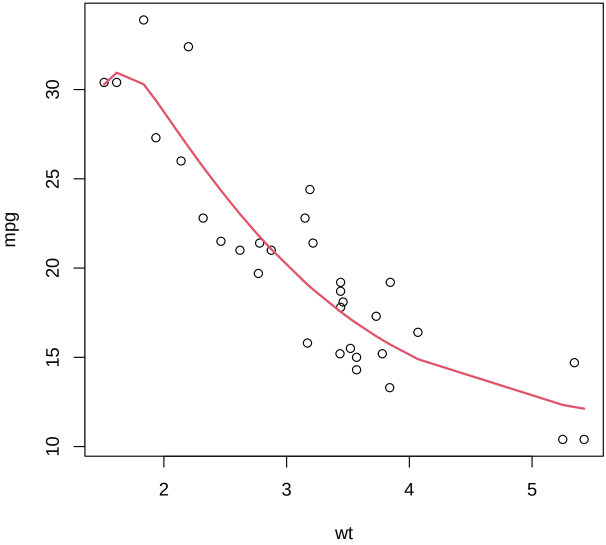
```
> red <- nnet (mpg ~ wt, mtcars, size = 100,
+               linout = TRUE,
+               trace = FALSE, maxit = 1000)
> red $ value
[1] 10.57239
> red <- nnet (mpg ~ wt, mtcars, size = 100,
+               linout = TRUE,
+               trace = FALSE, maxit = 1e6)
> red $ value
[1] 8.92599
```



mpg
10
15
20
25
30
wt
2
3
4
5

Regresión

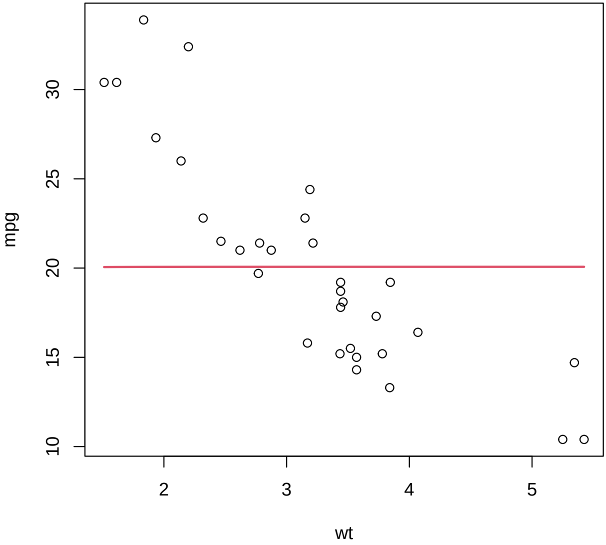
```
> red <- nnet (mpg ~ wt, mtcars, size = 2,
+              linout = TRUE,
+              decay = 0.001)
# weights:  7
initial  value 13949.841642
iter  10 value 1126.430780
iter  20 value 1126.383233
iter  30 value 276.940896
iter  40 value 206.158599
iter  50 value 203.596564
iter  60 value 203.563142
iter  70 value 203.430857
iter  80 value 203.030104
iter  90 value 198.182084
iter 100 value 197.436411
final  value 197.436411
stopped after 100 iterations
```



mpg
10
15
20
25
30
wt
2
3
4
5

Regresión

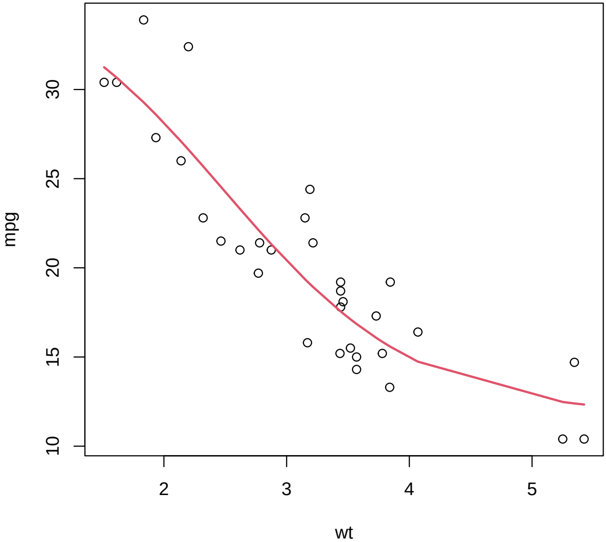
```
> red <- nnet (mpg ~ wt, mtcars, size = 2,
+              linout = TRUE,
+              decay = 0.1)
# weights:  7
initial  value 14090.158308
iter  10 value 1144.205367
iter  20 value 1143.164155
final  value 1143.101551
converged
```



mpg
wt
10
15
20
25
30
2
3
4
5

Regresión

```
> red <- nnet (mpg ~ wt, mtcars, size = 2,
+              linout = TRUE,
+              decay = 0.01)
# weights:  7
initial  value 14636.004583
iter  10 value 919.679895
iter  20 value 224.429971
iter  30 value 218.452659
iter  40 value 218.144572
iter  50 value 218.127501
final  value 218.127381
converged
```



10
15
20
25
30
mpg
2
3
4
5
wt

Decaimiento de pesos

- pretende evitar óptimos locales al ajustar los w_{ij}
- ajustar $E + \lambda \sum_{i,j} w_{ij}^2$
- tiene sentido si $0 \lessapprox x, y \lessapprox 1$
- se aconseja $0{,}0001 \lessapprox \lambda \lessapprox 0{,}01$

Recomendaciones

- tipificar / normalizar / rescalar las variables
- validación cruzada

```
> valcruz <- function (numneur, decai, partes=10)
+ {
+     iparte <- sample (rep (1:partes,
+                           length.out = nrow(mtcars)))
+     mean (sapply (1:partes,
+           function (i)
+           {
+               red <- nnet (mpg ~ wt,
+                            mtcars[iparte!=i,],
+                            linout = TRUE,
+                            size = numneur,
+                            decay = decai,
+                            trace = FALSE)
+               mean ((predict (red,
+                               mtcars[iparte==i,]) -
+                      mtcars$mpg[iparte==i]) ^ 2)
+           }))
+ }
```

```
> valcruz (  2, 0.001)
[1] 10.68111
> valcruz ( 10, 0.001)
[1] 10.90966
> valcruz (100, 0.001)
[1] 14.12828
> mean (reg $ residuals ^ 2)
[1] 8.697561
```

```
> valcruzreg <- function (partes=10)
+ {
+     iparte <- sample (rep (1:partes,
+                            length.out = nrow(mtcars)))
+     mean (sapply (1:partes,
+             function (i)
+             {
+                 reg <- lm (mpg ~ wt,
+                            mtcars[iparte!=i,])
+                 mean ((predict (reg,
+                                 mtcars[iparte==i,]) -
+                        mtcars$mpg[iparte==i]) ^ 2)
+             }))
+ }
```

```
> valcruzreg ()
[1] 11.051
> mean (reg $ residuals ^ 2)
[1] 8.697561
```

```
> mtcars <- data.frame (scale (mtcars))
> mean (lm(mpg~wt,mtcars) $ residuals ^ 2)
[1] 0.2394432
> valcruz (2, 0.001)
[1] 0.2381019
> valcruz (2, 0.001)
[1] 0.2678433
> valcruz (2, 0.001)
[1] 0.2424257
```

Bibliografía

- `https://cran.r-project.org/view=MachineLearning`
- Ripley B.; 1996; Pattern recognition and neural networks; Cambridge University Press
- Venables W., Ripley B.; 2002; Modern applied statistics with S; Springer