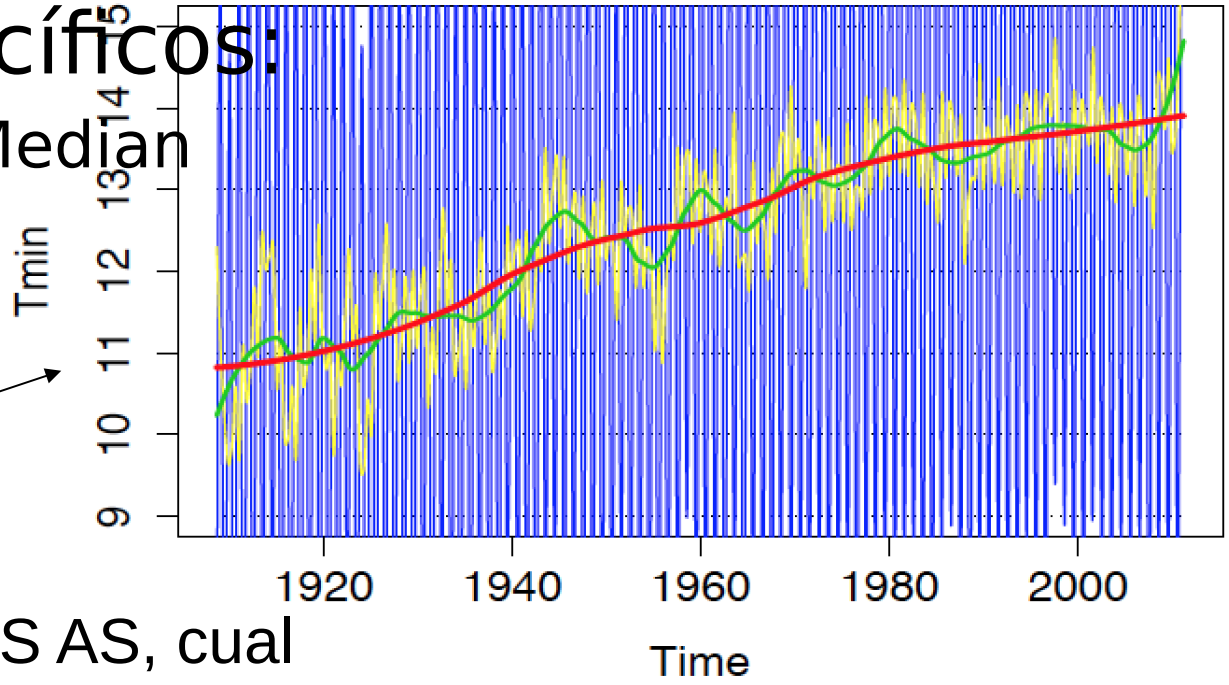
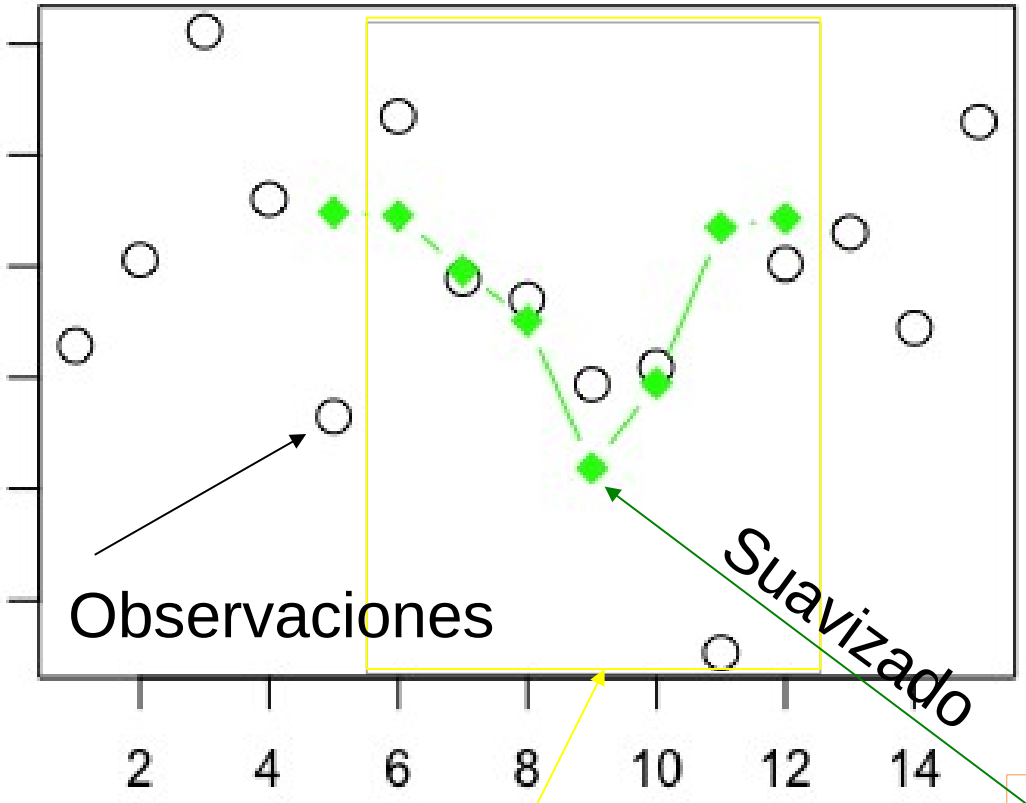
Regresión No Paramétrica
Scatterplot Smoothing

- Estima de manera No Paramétrica (no suponen un modelo (finito) paramétrico determinado) la relación entre la Y y la X.
- Métodos específicos:
  - Running Mean/Median
  - Running Line
  - LOESSS
  - Splines



100 años de Tmin en BS AS, cual es el verdadero comportamiento ?

Running Mean



Observaciones

Suavizado

Variable Respuesta

Error

$$y = f(x) + \epsilon$$

Función "suave"

Variable Explicativa

$$S(x_i) = \sum_{j \in N(x_i)} (y_j)/n_i$$

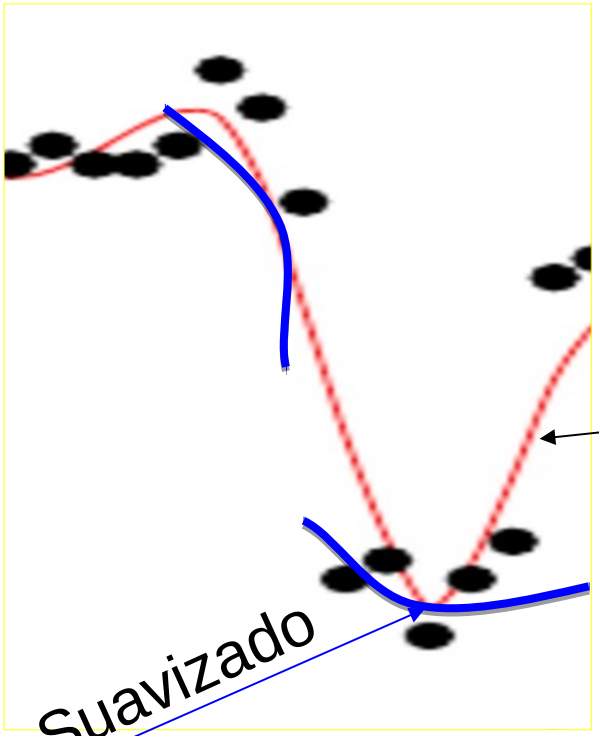
Ventana de 2k+1 elementos (k=3)

$$N(x_i) = \{\max(i-k,1), \ldots, i-1, i, i+1, \ldots, \min(i+k,n)\}$$

Datos ordenados $\longrightarrow$ $x_1 < x_2 < \ldots < x_n$

LOWESS Robusto (Cleveland 1979

LOWESS



Suavizado

Ventana fraccionaria

Ventana entera

$$0 < f \leqslant 1 \quad \text{let } r \text{ be } fn \text{ rounded}$$

Evaluable en todo punto x

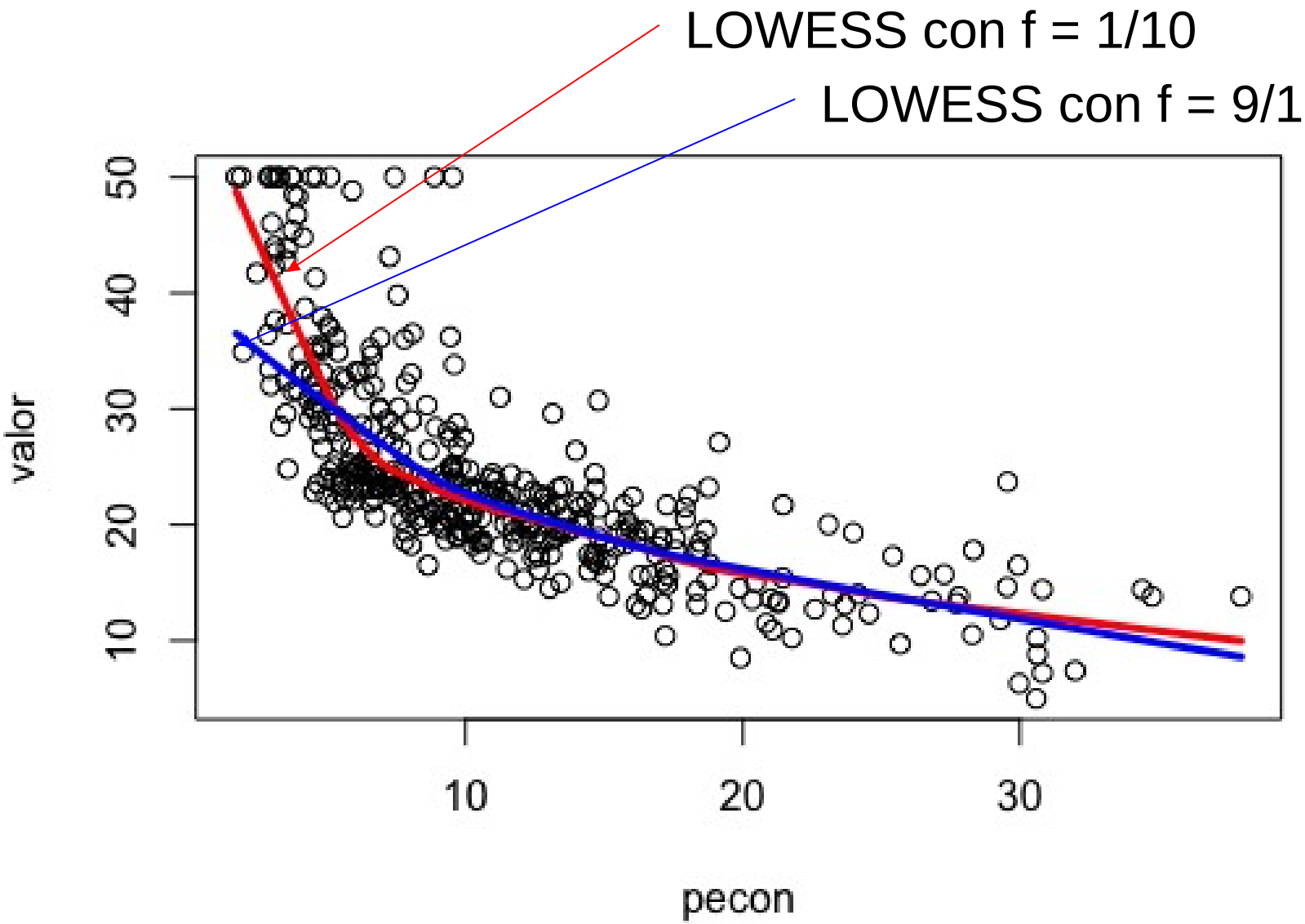
Función de pesos centrada en x_i decreciente que se hace 0 a los r puntos de x_i

Polinomio de grado d

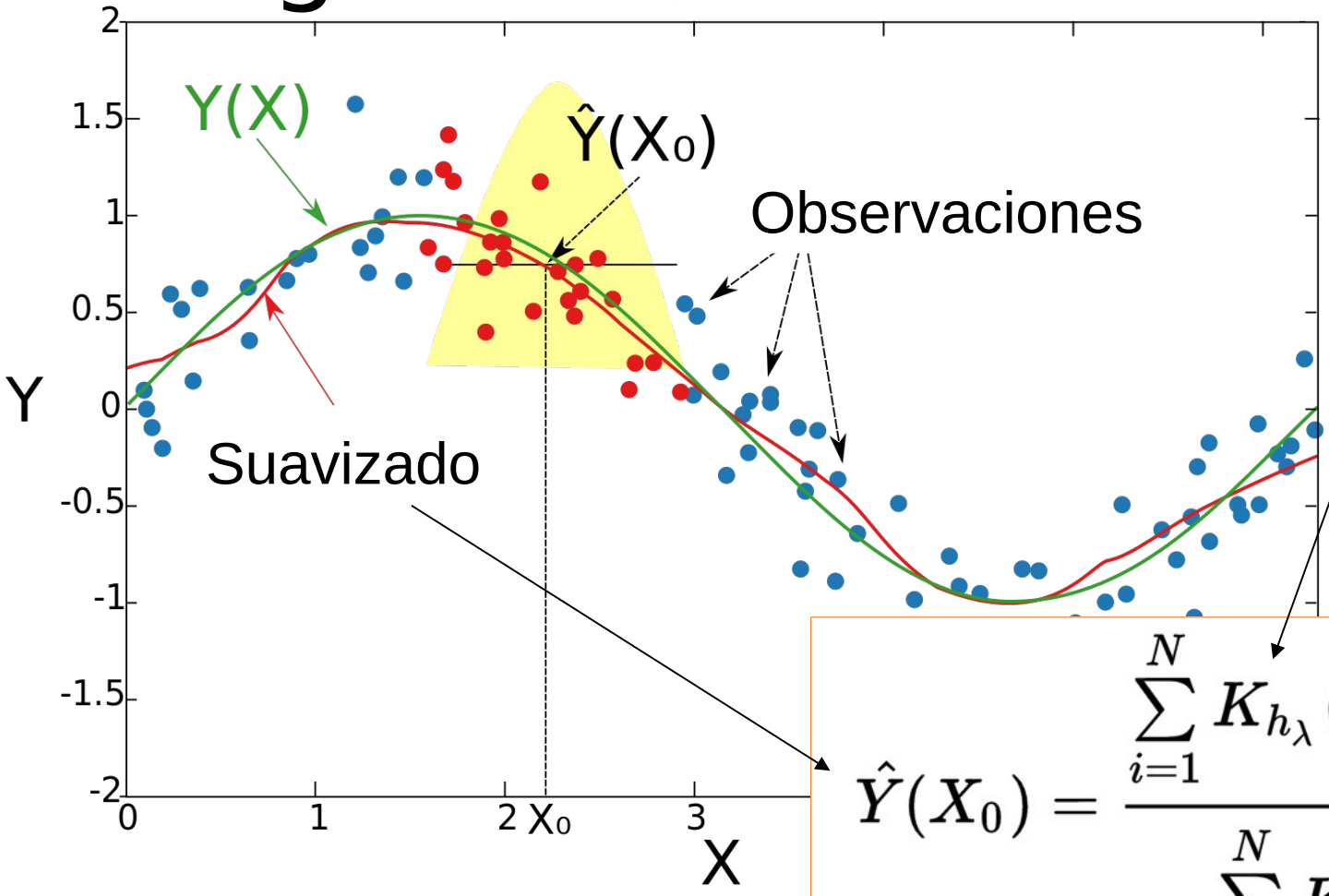
$$\hat{y}_i = \sum_{j=0}^{d} \hat{\beta}_j(x_i) x_i{}^j$$

$$\hat{\beta}_j(x_i) = \text{ARGMIN} \sum_{k=1}^{n} w_k(x_i)\,(y_k - \beta_0 - \beta_1 x_k - \ldots - \beta_d x_k{}^d)^2$$

Ejemplo: Valor Vs. Pecon



Regresión/Suavizado Kernel



Función Kernel que pesa a las obs. por cercanía

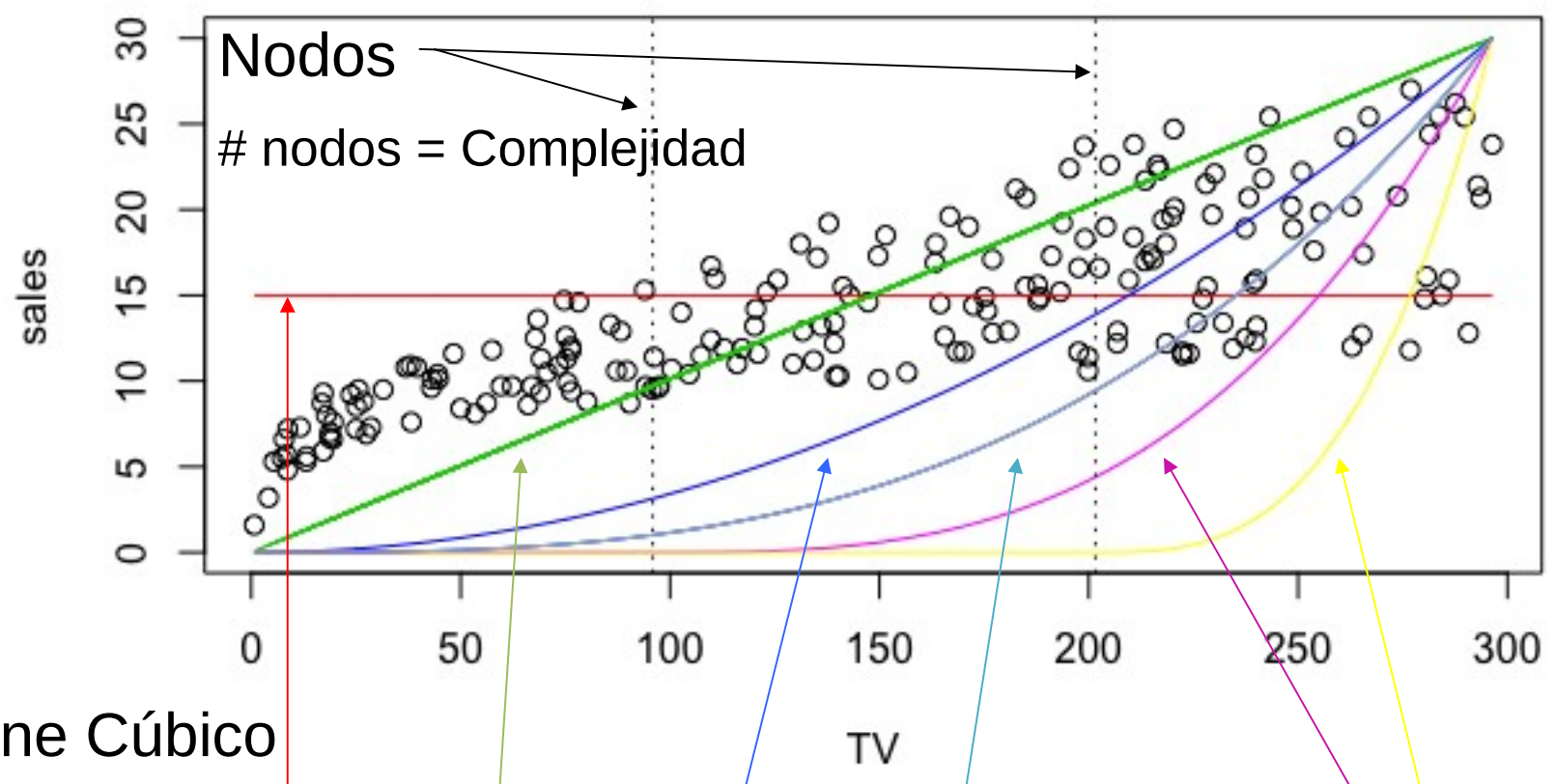
$$\hat{Y}(X_0) = \frac{\sum_{i=1}^{N} K_{h_\lambda}(X_0, X_i) Y(X_i)}{\sum_{i=1}^{N} K_{h_\lambda}(X_0, X_i)}$$

$$K_{\boldsymbol{\lambda}}(x^*, x_i) = \exp\left(-\frac{(x^* - x_i)^2}{\boldsymbol{\lambda}}\right)$$

Parámetro de suavizado

olines Cúbicos Truncados

Parametros =
3 * 4 – 2 * 3 = 6



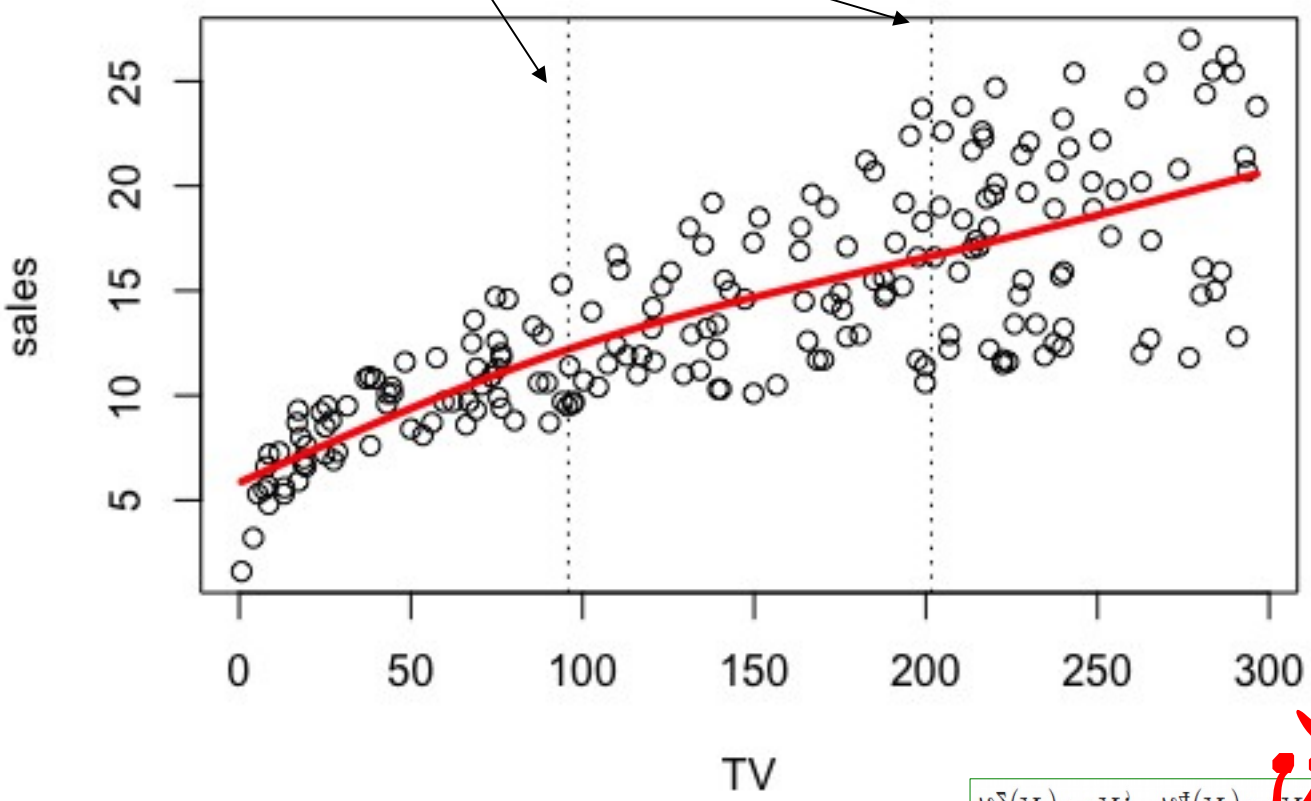
Spline Cúbico

$$S(x) = \beta_o + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{k} \gamma_j (x - \xi_j)_+^3.$$

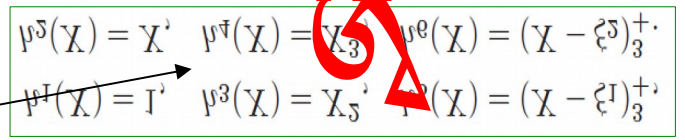
Resultado del Regression Spline

Son los splines de menor grado cuya potencial falta de continuidad en los nodos NO es visualmente detectable

Continuos hasta la 2da derivada



Se trbaja con una base equivalente a la “Truncada” que se llama B-splines



Splines Cúbicos Suavizados

Smooth.spline

- Son smoothers de tipo “piecewise polynomials” definidos sobre intervalos contiguos de todas las X,’s (tantos nodos como observaciones).
- Resultan de ajustar una base polinomial de grado 3 con 2da derivada continua en los nodos.
- Son una alternativa muy difundida y son (de alguna manera) óptimos (entre FUN con 2da der. continua).
- Solucionan la siguiente función de ajuste con penalización:

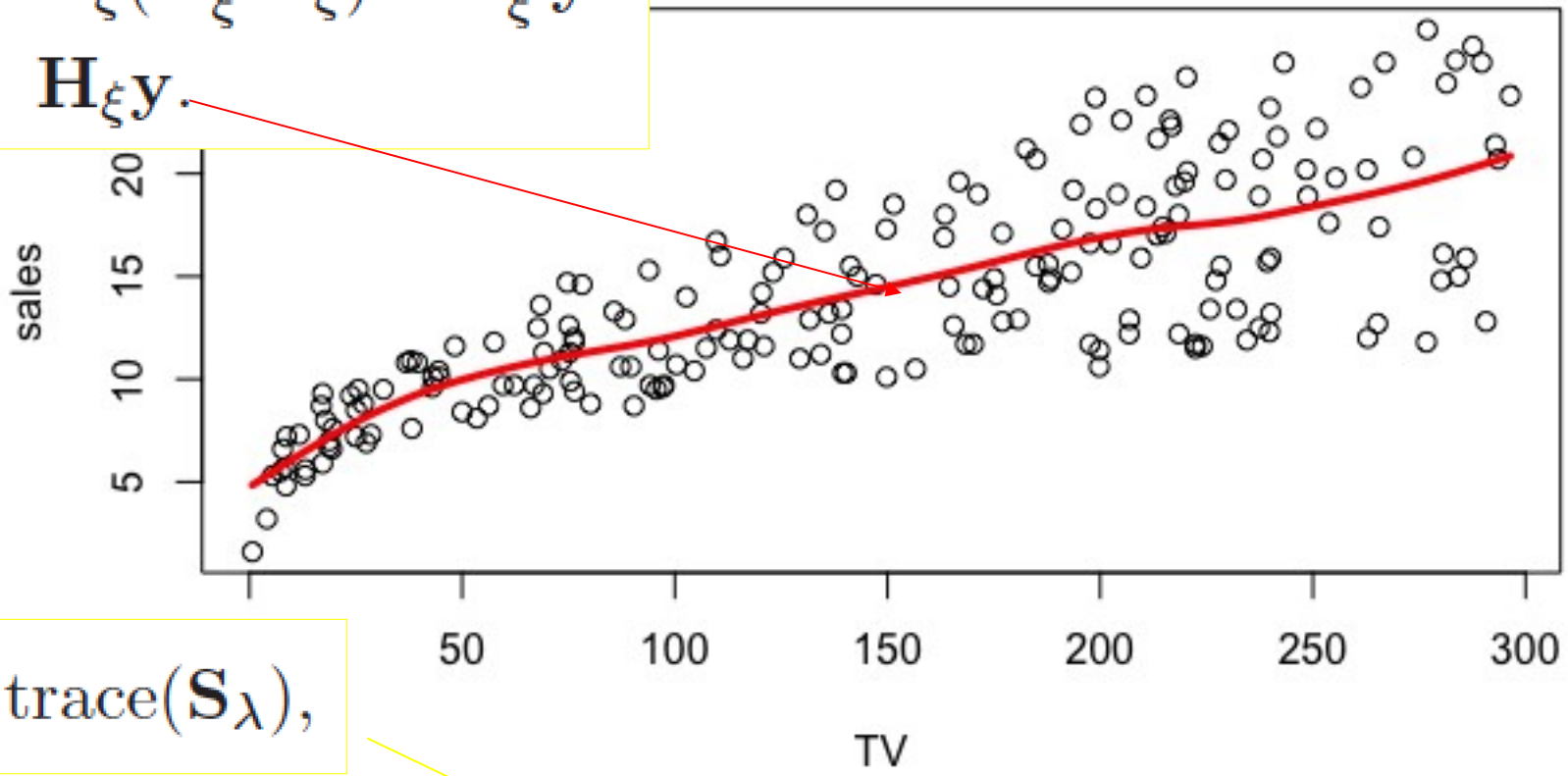
$$\sum_{i=1}^{n}(y_i - S(x_i))^2 + \lambda \int (S''(x))^2 dx$$

Falta de Ajuste a las Observaciones

Penalización por Curvatura

Ejemplo

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{B}_\xi(\mathbf{B}_\xi^T\mathbf{B}_\xi)^{-1}\mathbf{B}_\xi^T\mathbf{y} \\ &= \mathbf{H}_\xi\mathbf{y}.\end{aligned}$$



$$\mathrm{df}_\lambda = \mathrm{trace}(\mathbf{S}_\lambda),$$

```
smooth.spline(x = TV, y = sales, spar = 9/10)

Smoothing Parameter  spar= 0.9  lambda= 0.001998161
Equivalent Degrees of Freedom (Df): 7.221166
Penalized Criterion: 1937.016
GCV: 10.83938
```

Modelos Aditivos

gam

- Extienden y generalizan el modelo de regresión lineal clásico.
- Ajustan/estiman relaciones suaves entre la Y y las X's.
- No asumen linealidad entre parámetros y variables.
- Son la base de técnicas más complejas:
  - Projection Pursuit Regression
  - Artificial Neural Networks

$$\mathrm{E}(Y|X_1, X_2, \ldots, X_p) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$

El Modelo y la Estimación

$\sum_1^N f_j(x_{ij}) = 0 \ \forall j$

Modelo

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \varepsilon$$

Error con media = 0

Función de Pérdida a Minimizar

$$\mathrm{PRSS}(\alpha, f_1, f_2, \ldots, f_p) = \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j$$

Falta de Ajuste

Penalización

El Algoritmo “Backfitting”
Leo Breiman y Jerome Friedman (1985)

1. Initialize: $\hat{\alpha} = \frac{1}{N}\sum_1^N y_i,\ \hat{f}_j \equiv 0, \forall i, j.$

2. Cycle: $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots, p, \ldots,$

$$\hat{f}_j \leftarrow \mathcal{S}_j\left[\{y_i - \hat{\alpha} - \sum_{k\neq j} \hat{f}_k(x_{ik})\}_1^N\right],$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N}\sum_{i=1}^N \hat{f}_j(x_{ij}).$$

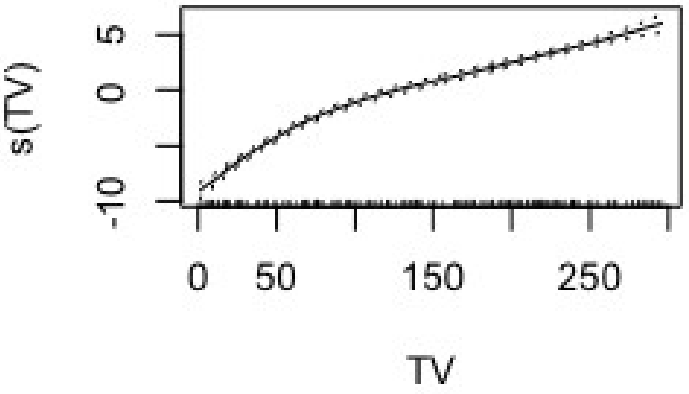
Smoothing Cubic Spline

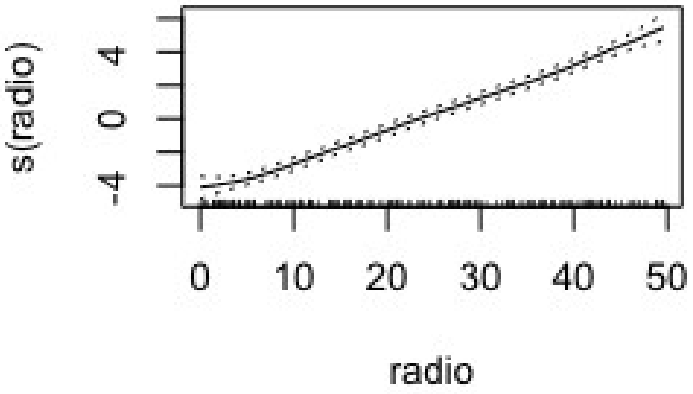
Residuos generados por no usar la var j-ésima

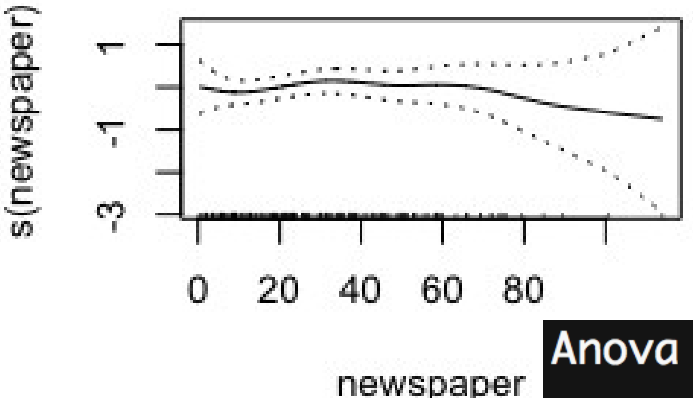
Re-centrado

until the functions $\hat{f}_j$ change less than a prespecified threshold.

Ejemplo de GAM








```
Anova for Nonparametric Effects
             Npar Df  Npar F     Pr(F)
(Intercept)
s(TV)              3 25.9057 4.641e-14 ***
s(radio)           3  1.5953    0.1920
s(newspaper)       3  1.2851    0.2808
```

```
Anova for Parametric Effects
             Df Sum Sq Mean Sq   F value Pr(>F)
s(TV)         1 3321.0  3321.0 1636.9921 <2e-16 ***
s(radio)      1 1672.7  1672.7  824.5047 <2e-16 ***
s(newspaper)  1    0.0     0.0    0.0019  0.965
Residuals   187  379.4     2.0
```

Modelos Lineales Mixtos

lme4

- Permiten modelar la falta de independencia de las observaciones (frailty effect).
- Ahorran grados de libertad (parsimonia).
- Modelan situaciones REALES, NO EXPERIMENTALES (e.j. Diseño de experimentos).
- Contemplan la posibilidad de anidamiento (nesting) de las observaciones.
- A vecas, son el modelo CORRECTO !

El Concepto detras de LME

H_0: $\alpha_1 = \alpha_2 = \cdots = \alpha_a$

H_a: At least one inequality

μ: A constant, the mean of all possible experiments using the a designated treatments

α_i: A constant for the ith treatment group, the deviation from the mean due to the ith treatment: $\sum_i \alpha_i = 0$

ε_{ij}: A random effect containing all uncontrolled sources of variability. The ε_{ij}'s are IND $(0, \sigma^2)$, that is, they are normally distributed with a mean of zero and a variance σ^2 and they are independent of each other and of the α_i's.

H_0: $\sigma_A^2 = 0$

H_a: $\sigma_A^2 > 0$

μ: A constant, the population mean for all experiments involving all possible treatments of the type being considered

α_i: A constant for the ith treatment group, a random deviation from the population mean. The α_i's are normal, with $E(\alpha_i) = 0$ and $V(\alpha_i) = \sigma_A^2$

ε_{ij}: Same as for FEM

Obs.

Trat.

Indiv.

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$$

$$i = 1, 2, \ldots, a$$

$$j = 1, 2, \ldots, n$$

Dowdy, S., Wearden, S., & Chilko, D. (2011). Statistics for research (Vol. 512). John Wiley & Sons.

Ejemplo: Rindes de Cultiv

$$R_{i,j} = \mu + Si_{i,j} + Ca_i + Su_j + Co_j + Lo_j$$

Donde:

$R_{i,j}$ es el rinde del año (campaña) i en el lote j

μ es el rinde promedio

$Si_{i,j}$ es el efecto fijo del tipo de siembra del año i en el lote j

Ca_i es el efecto fijo del año i

Su_j es el efecto fijo del suelo en el lote j

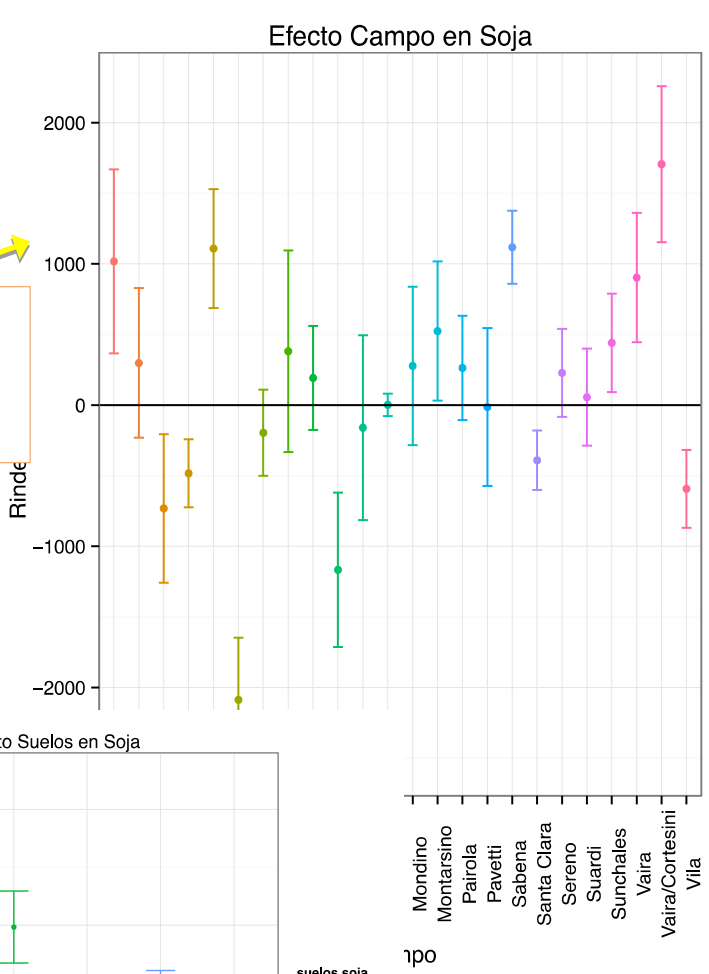
Co_j es el efecto fijo del campo del lote j

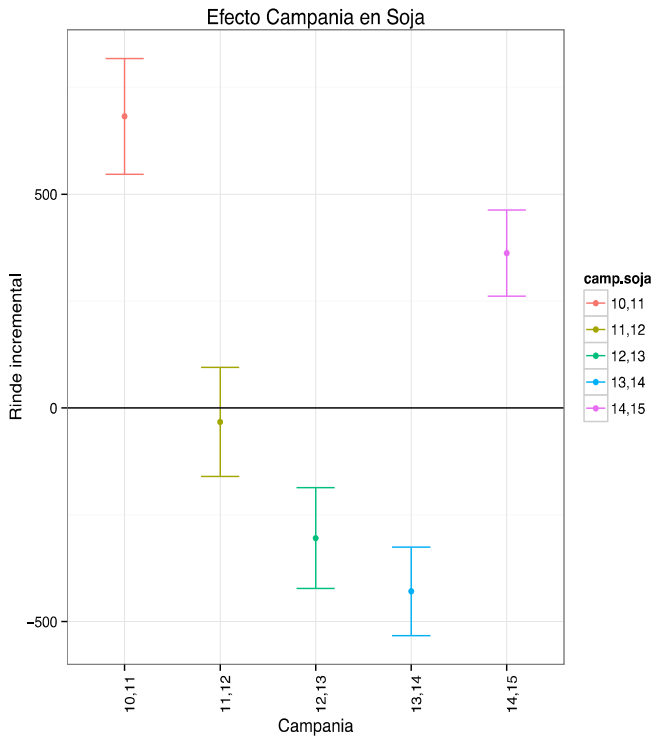
$Lo_{i,j}$ es el efecto aleatorio del lote j

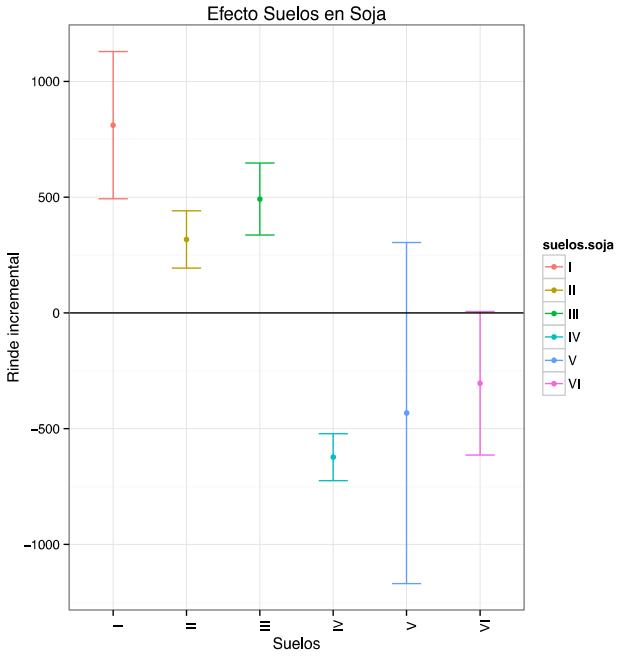
Siembre (primera y segunds+tardía)
Campaña
Tipo de suelo (categorías simplificadas: I,II,III,IV,V y VI)
Campo (variable de 24 categóricas)
Lote (variable de 143 categorías)

Estimaciones

$$R_{i,j} = \mu + Si_{i,j} + Ca_i + Su_j + Co_j + Lo_j$$





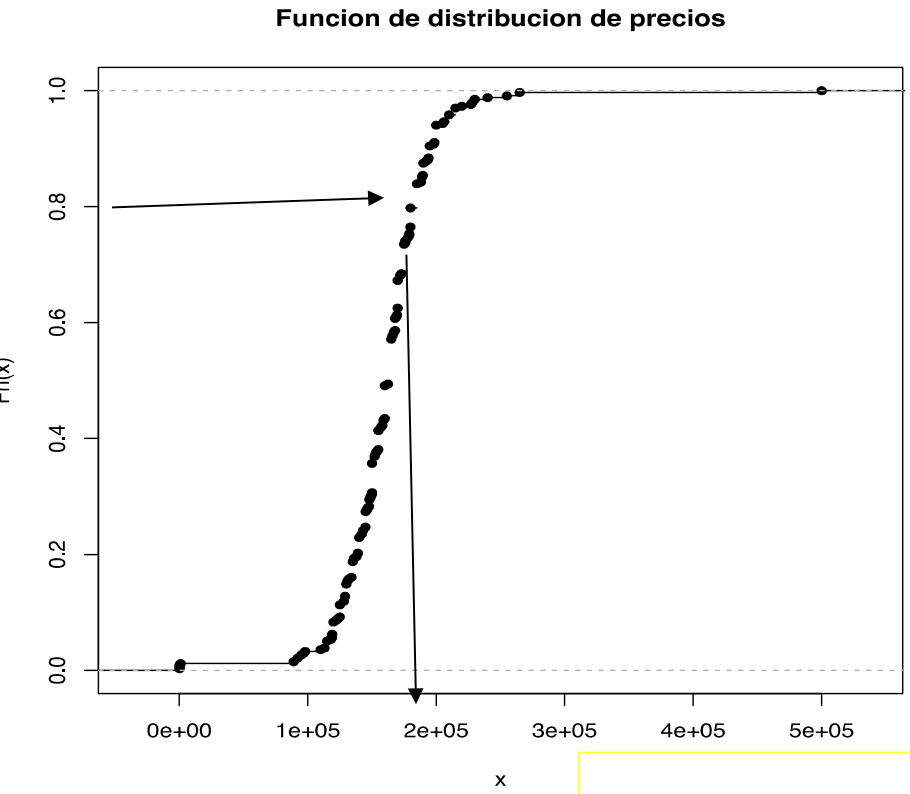


Quantile Regresion (QR

quantreg

- Estimador de la curva del Cuantil Condicional
- Es un método naturalmente robusto (regresión)
- Capta atributos generales de la distribución de Y (no sólo la centralidad)
- Permite generar facilmente bandas de predicción
- No supone homocedasticidad
- No asume supuestos distribucionales

El Cuantil No-Condicional



Función de distribución d
la variable a explicar (Y)

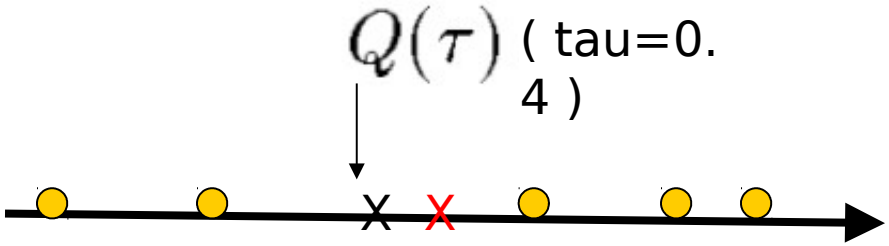
$$F(y) = \text{Prob}(Y \leq y)$$

$$Q(\tau) = \inf\{y : F(y) \geq \tau\}$$

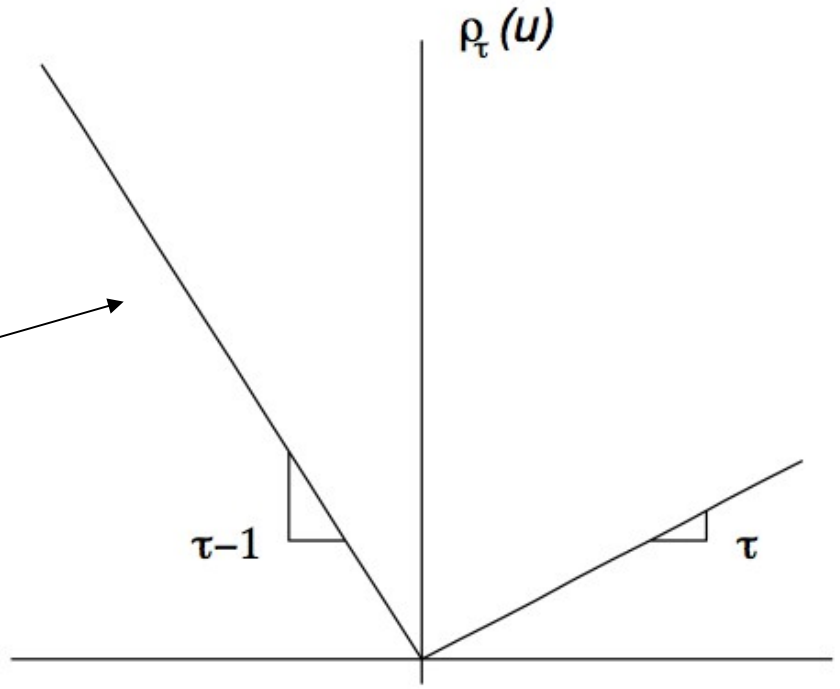
Cuantil tau de la

Definición de cuantil (empírico) desde la optimización

$$Q(\tau) = \operatorname{argmin}_{Q} \sum \rho_\tau(y_i - Q) \quad 0 < \tau < 1$$



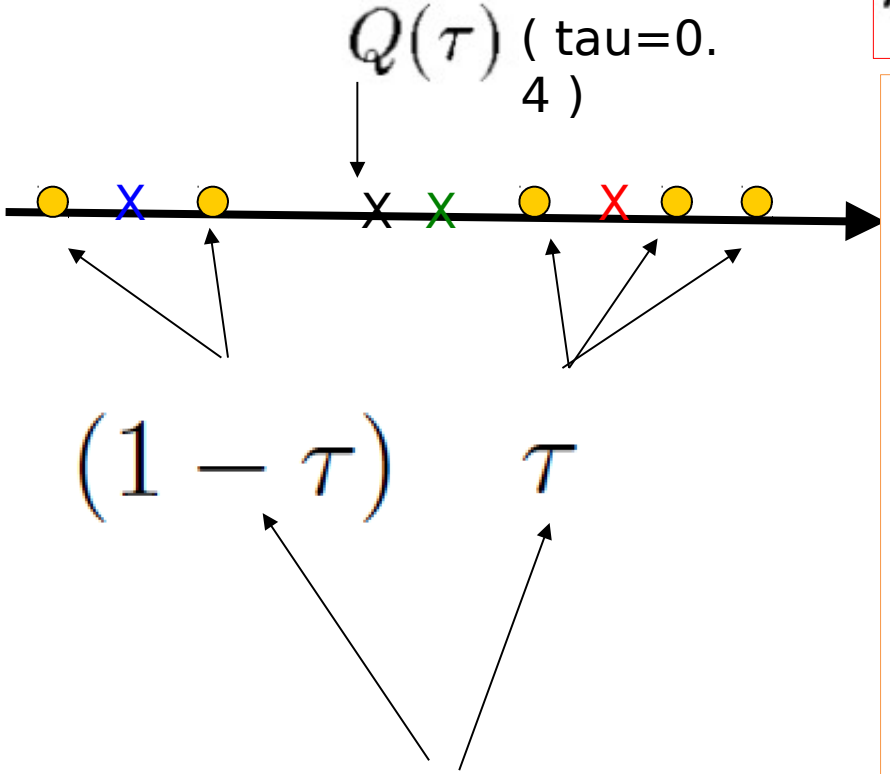
$$\rho_\tau(u) = u(\tau - I(u < 0))$$



La intuición de porque funciona

$$V = \sum \rho_\tau(y_i - Q) \quad 0 < \tau < 1$$

$$\rho_\tau(u) = u(\tau - I(u < 0))$$

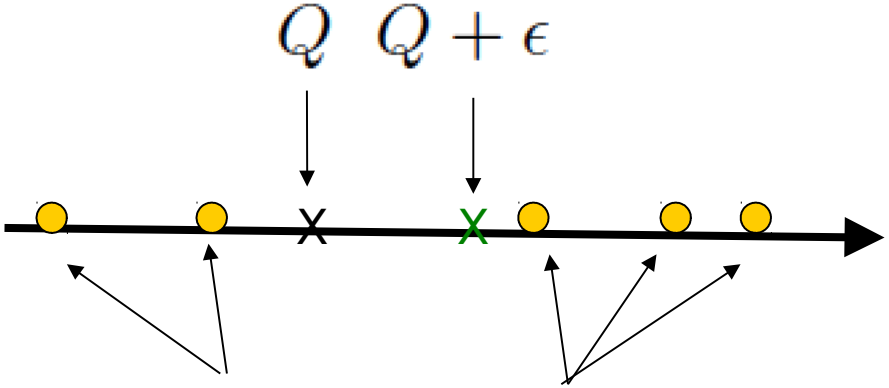


$$\begin{aligned}
V &= \sum_{i=1}^{n}(y_i - Q)(\tau - I(y_i < Q)) = \\
&= \sum_{i=1}^{n} u_i(\tau - I(u_i < 0)) \\
&= \sum_{i/y_i \geq Q} u_i\tau + \sum_{i/y_i < Q} u_i(\tau - 1) \\
&= \sum_{i \in D} u_i\tau + \sum_{i \in I} u_i(\tau - 1) \\
&= \sum_{i \in D} d_i\tau + \sum_{i \in I} d_i(1 - \tau)
\end{aligned}$$

$$d_i = abs(u_i)$$

uando se mejora la función ?

voy de Q a $Q+\epsilon$, con $\epsilon > 0$



$$(1-\tau) \quad \tau$$

$$\begin{aligned} mejora(V) &= -\Delta V = \epsilon * (\#D) * \tau - \epsilon * (\#I) * (1-\tau) \\ &= \epsilon * (\#D/N) * N\tau - \epsilon * (\#I/N) * N(1-\tau) \end{aligned}$$

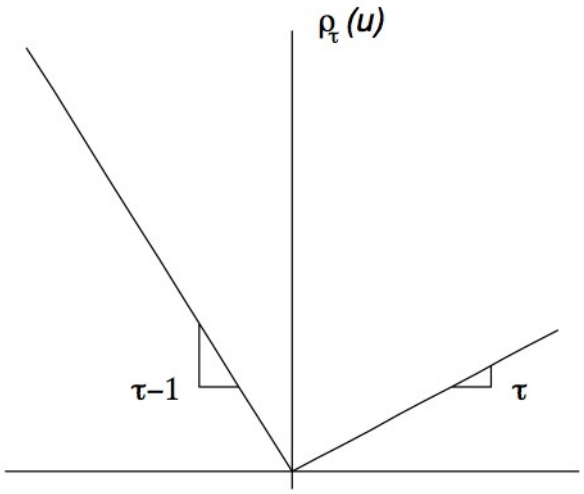
$$mejora(V) > 0 \Leftrightarrow (\#D/N) * N\tau > (\#I/N) * N(1-\tau)$$

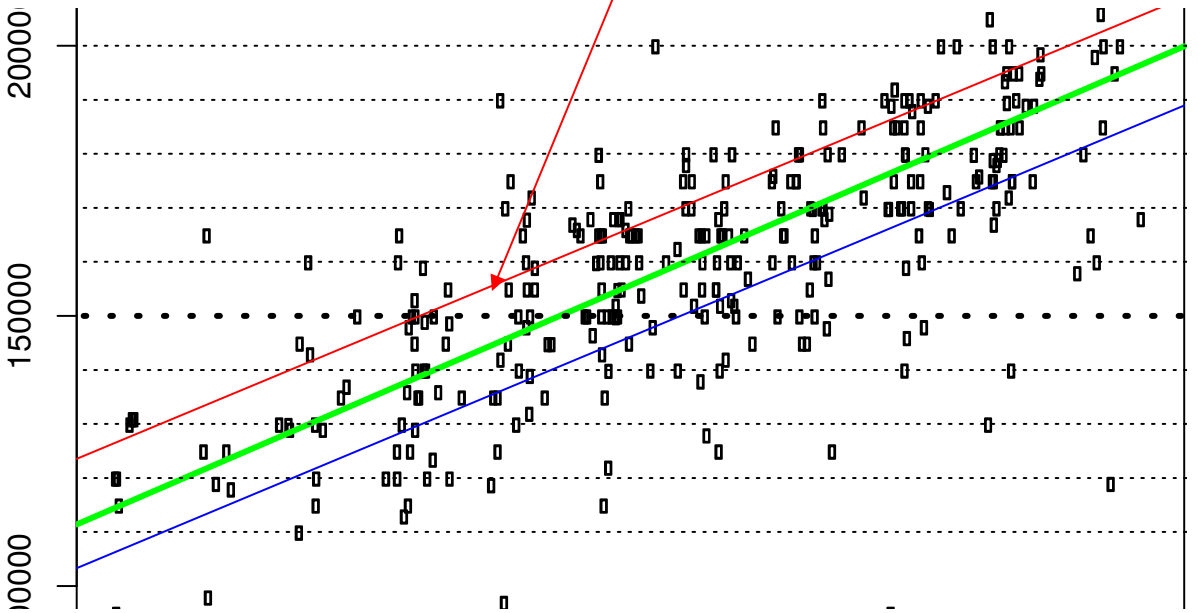
o sea

$$mejora(V) > 0 \Leftrightarrow \#D/\#I > (1-\tau)/\tau$$

Quantile Regression

$$\hat{\beta}(\tau) = \text{argmin}_{\beta \in \mathbb{R}^p} \sum \rho_\tau (y_i - x'\beta)$$





$$\rho_\tau(u) = u(\tau - I(u < 0))$$

Ejemplo de QR

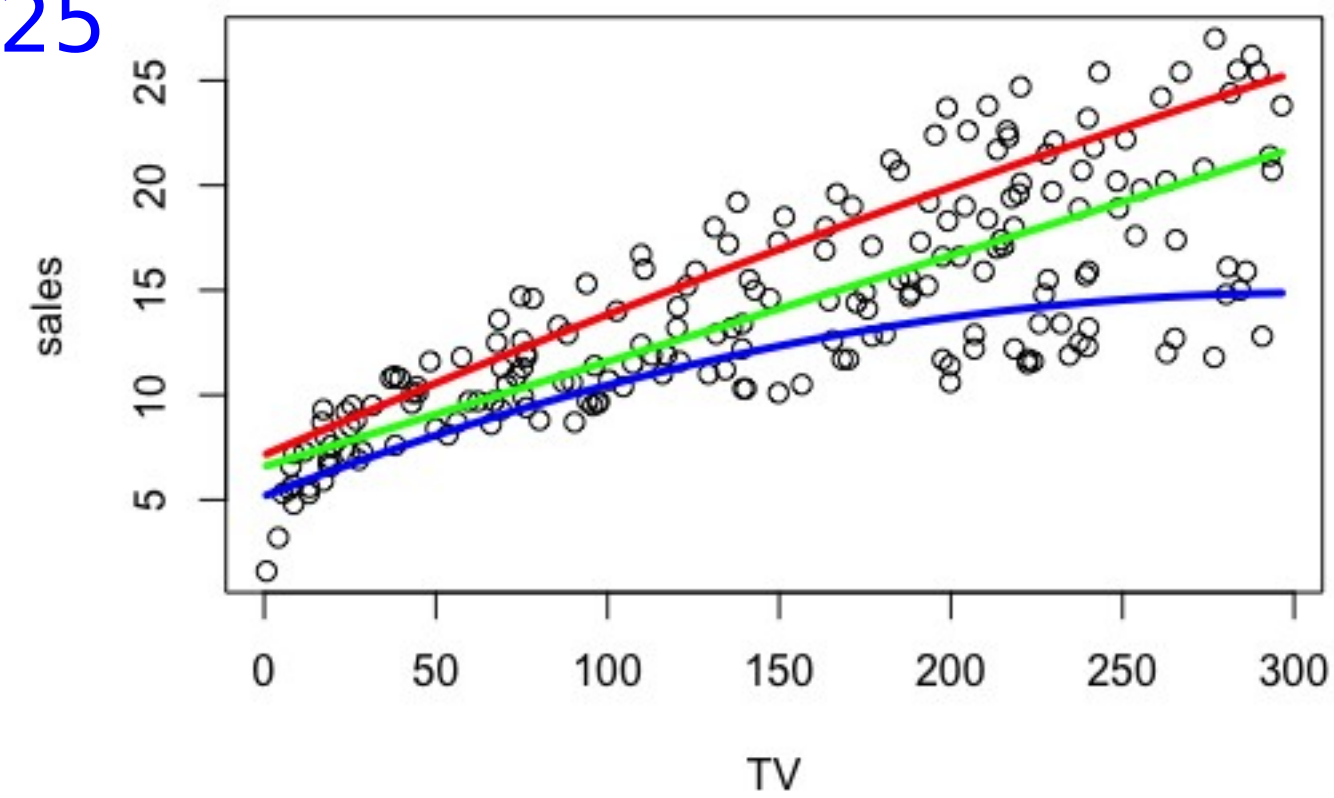
Estimación de curvas de :

Percentil 75

Percentil 50

Percentil 25

```
ajus.q1<-rq(sales~poly(TV,2),tau=0.25)
ajus.q2<-rq(sales~poly(TV,2),tau=0.75)
ajus.qm<-rq(sales~poly(TV,2),tau=0.5)
```



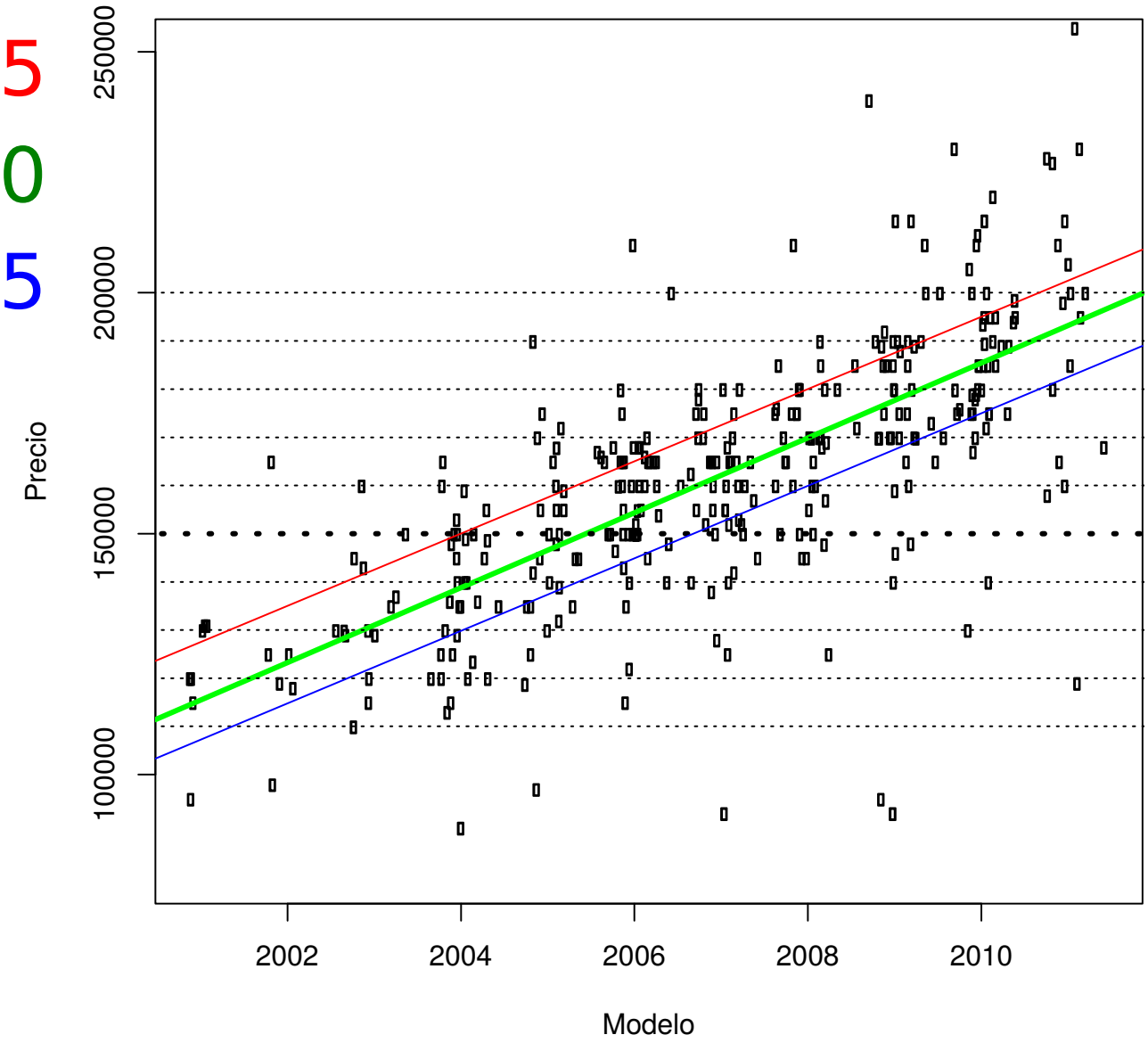
Un Ejemplo Real

Estimación de curvas de :

Percentil 75

Percentil 50

Percentil 25



Projection Pursuit Regression

PPR

- Técnica supervizada que aproxima una respuesta univariada como suma de funciones continuas de proyecciones lineales de las covariables predictoras.
- Es un mejora/generalización sobre el modelo lineal de regresión.
- Es un aproximador universal.
- Como técnica supervizada puede verse como una versión simplificada de una Red Neuronal.

Friedman, Jerome H., and Werner Stuetzle. "Projection pursuit regression." Journal of the statistical Association 76.376 17-823.

- propuesta por Friedman y Tukey (1974)
- para regresión Friedman y Stuetzle (1981)
- usando splines Roosen y Hastie (1994)

Bsado en Material de M.E. Szretter

El Modelo

Idea central

extraer combinaciones lineales de las variables explicativas y luego modelar la respuesta como una función no lineal de estas combinaciones lineales

Modelo:

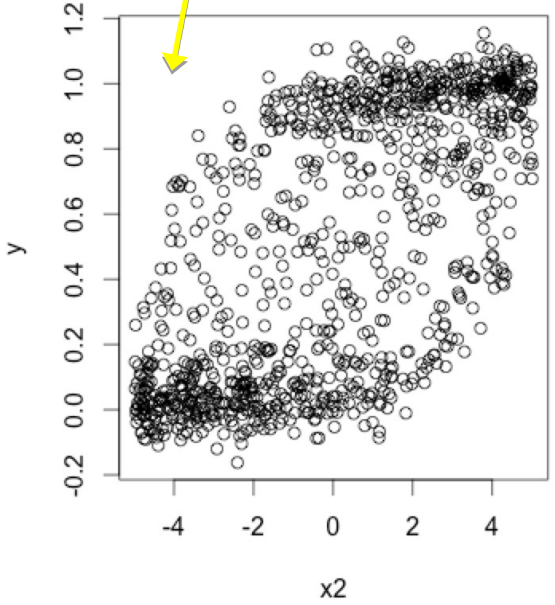
$$f(\mathbf{X}) = \sum_{m=1}^{M} g_m\left(\mathbf{w}_m^T \mathbf{X}\right)$$

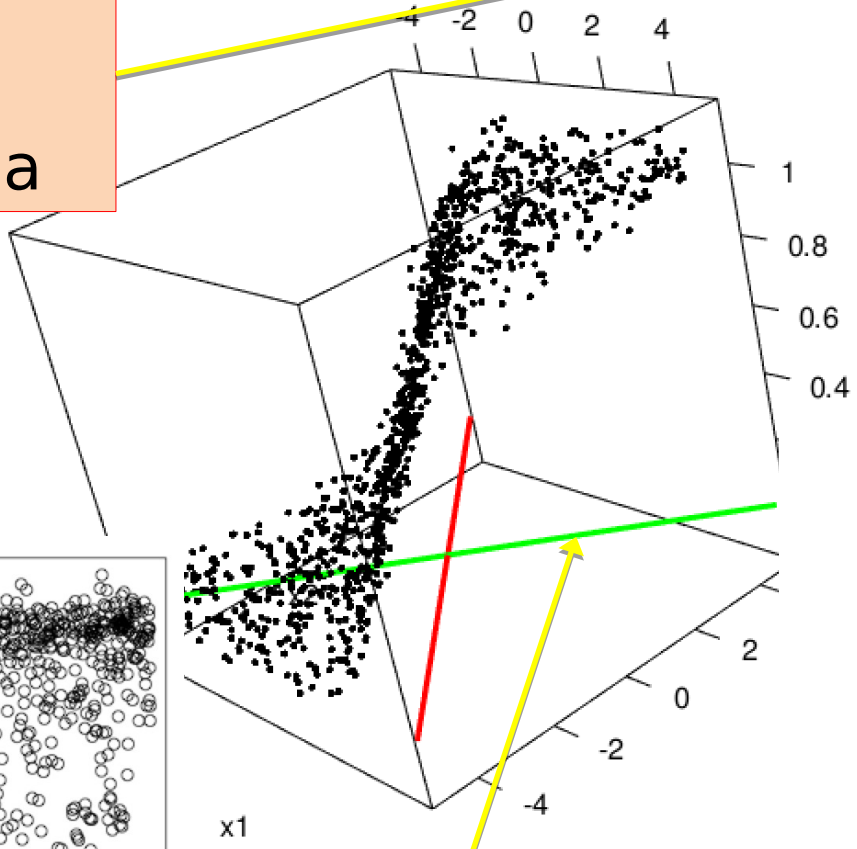
Función "ridge"

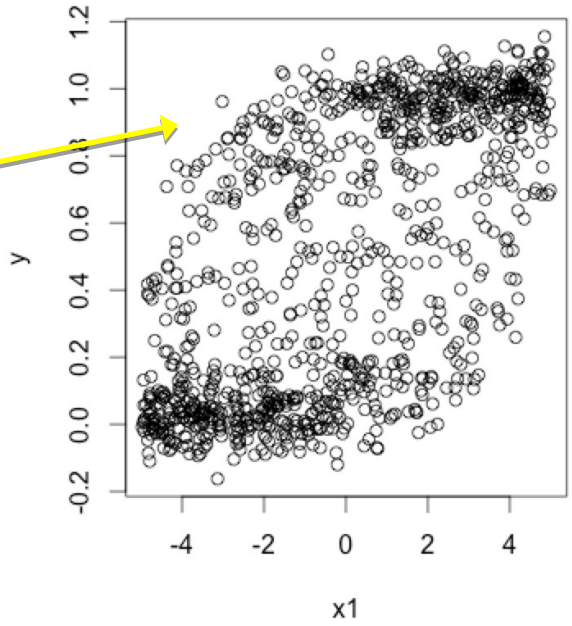
- donde $\mathbf{w}_m$ $(m = 1, \ldots, M)$ son vectores unitarios p-dimensionales de parámetros desconocidos
- y g_m son funciones no especificadas a estimar con los datos

La Intuición

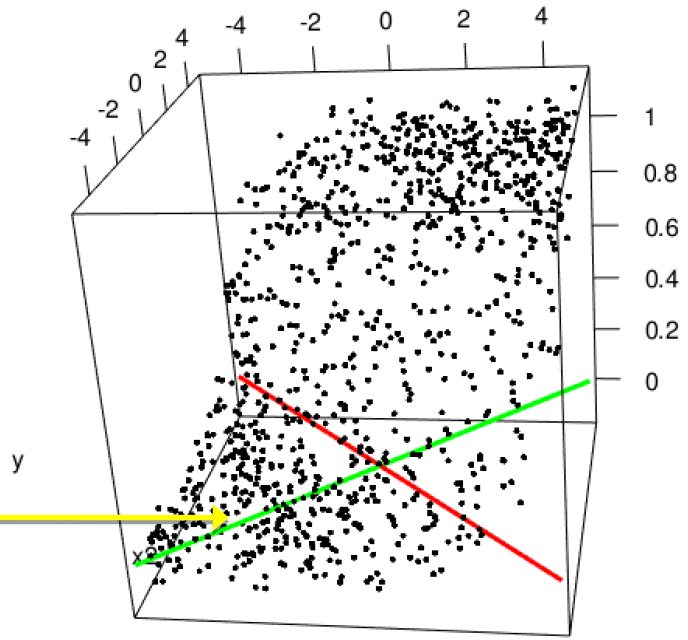
Escasa relación univariada





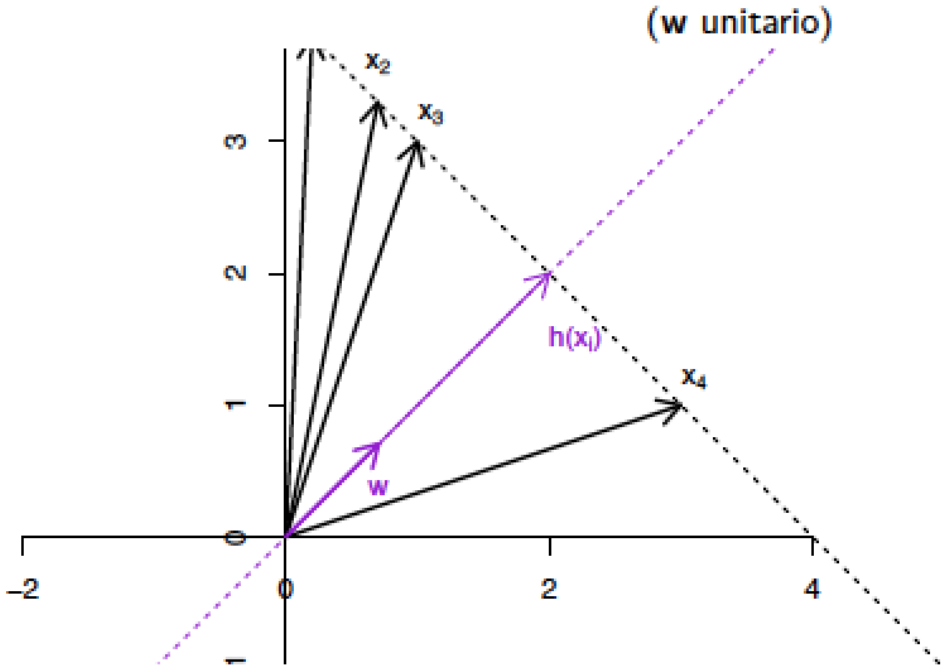


Dirección a ser hallada



Busqueda de la Dirección

La función $h : \mathbb{R}^p \to \mathbb{R}$, $h(\mathbf{X}) = \mathbf{w}^T\mathbf{x}$ es la proyección del vector $\mathbf{X}$



Buscamos $\mathbf{w_m}$ para que el modelo ajuste bien: Projection Pursuit

La función $g_m\left(\mathbf{w}_m^T\mathbf{X}\right)$ es constante en hiperplanos (ortogonales a w_m) y es unidimensional por lo que se sortea la maldición de la dimensionalidad

PPR como Aproximador Universal

Porque la operación de aplicar funciones no lineales a combinaciones lineales genera una gran cantidad de clases de modelos. Por ejemplo

$$X_1 X_2 = \frac{1}{2}\left[\left(\frac{X_1 + X_2}{\sqrt{2}}\right)^2 - \left(\frac{X_1 - X_2}{\sqrt{2}}\right)^2\right]$$

Si M se toma suficientemente **grande**, para una elección apropiada de g_m, el modelo PPR puede aproximar cualquier función de $\mathbb{R}^p$ pero...la interpretación del modelo es **difícil**

- PPR sirve para **predecir**
- Excepto, cuando $M = 1$ (Single Index Model)

Estimación

Dados $(\mathbf{X}_i, Y_i)$, $(i = 1, \ldots, n)$, ¿cómo ajustamos? Buscamos minimizar la función

$$\sum_{i=1}^{n} \underbrace{\left[Y_i - \sum_{m=1}^{M} g_m \left(\mathbf{w}_m^T \mathbf{X_i} \right) \right]}_{\text{residuo de la i-ésima observación}}^2 \quad (1)$$

Asumimos **w** fijo

Transformamos las observaciones: $V_i = \mathbf{w}_m^T \mathbf{X_i}$ Tenemos un problema de suavizado univariado $(V_i, Y_i)_i$ observaciones en $\mathbb{R}^2$ Ajustamos una regresión no paramétrica: smoothing spline o promedios móviles.

Asumimos g fija

Queremos minimizar (1) sólo en **w**. Resulta equivalente a un problema de regresión de mínimos cuadrados pesados

Los pasos para actualizar a g y **w** se iteran hasta converger

Ajuste de PPR: caso $M > 1$

Supongamos que hemos determinado los primeros $\widehat{g}_1, \ldots, \widehat{g}_{m-1}$ y $\widehat{\mathbf{w}}_1, \ldots, \widehat{\mathbf{w}}_{m-1}$. Sean

$$R_i = Y_i - \sum_{j=1}^{m-1} \widehat{g}_j \left(\widehat{\mathbf{w}}_j^T \mathbf{X} \right)$$

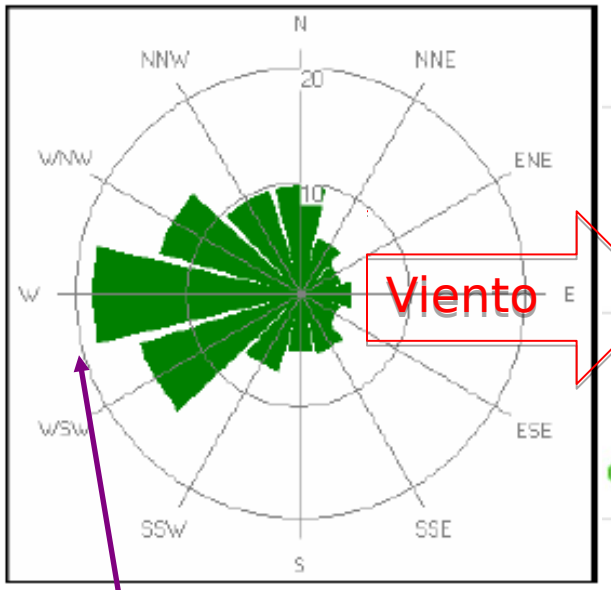
los residuos de esta aproximación. Le aplicamos el algoritmo anterior a los pares $(\mathbf{X}_i, R_i)$, para obtener $\widehat{g}_m$ y $\widehat{\mathbf{w}}_m$. El procedimiento se itera hasta que la mejora que se obtiene de la función objetivo es muy pequeña.
Muchos detalles de implementación:

- ¿qué técnica de suavizado se usa?
- ¿cómo se determina el parámetro de suavizado?
- estimación de M: validación cruzada o parte de la estrategia forward

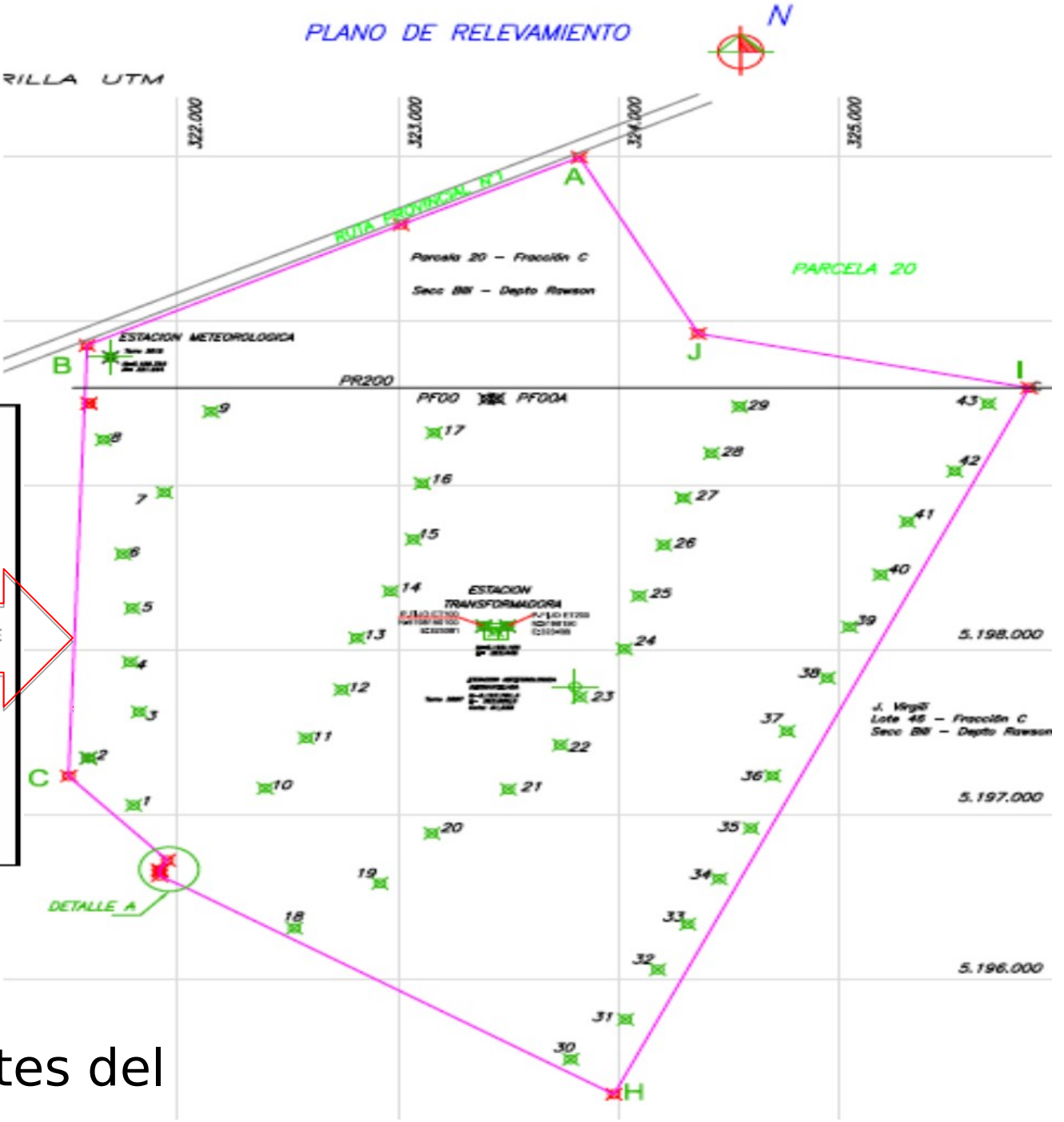
Ejemplo de PPR = Energía Eólica

Rawson: 43 molinos de 1.8MW

El parque eólico de Rawson



Vientos predominantes del oeste



$$KE = \frac{1}{2} \times M \times V^2$$

Air density

$$M = \rho \times \text{Vol} \qquad \text{Vol} = A \times D$$

Depth

Area

$$KE = \frac{1}{2} \times (\rho \times A \times D) \times V^2$$

Time

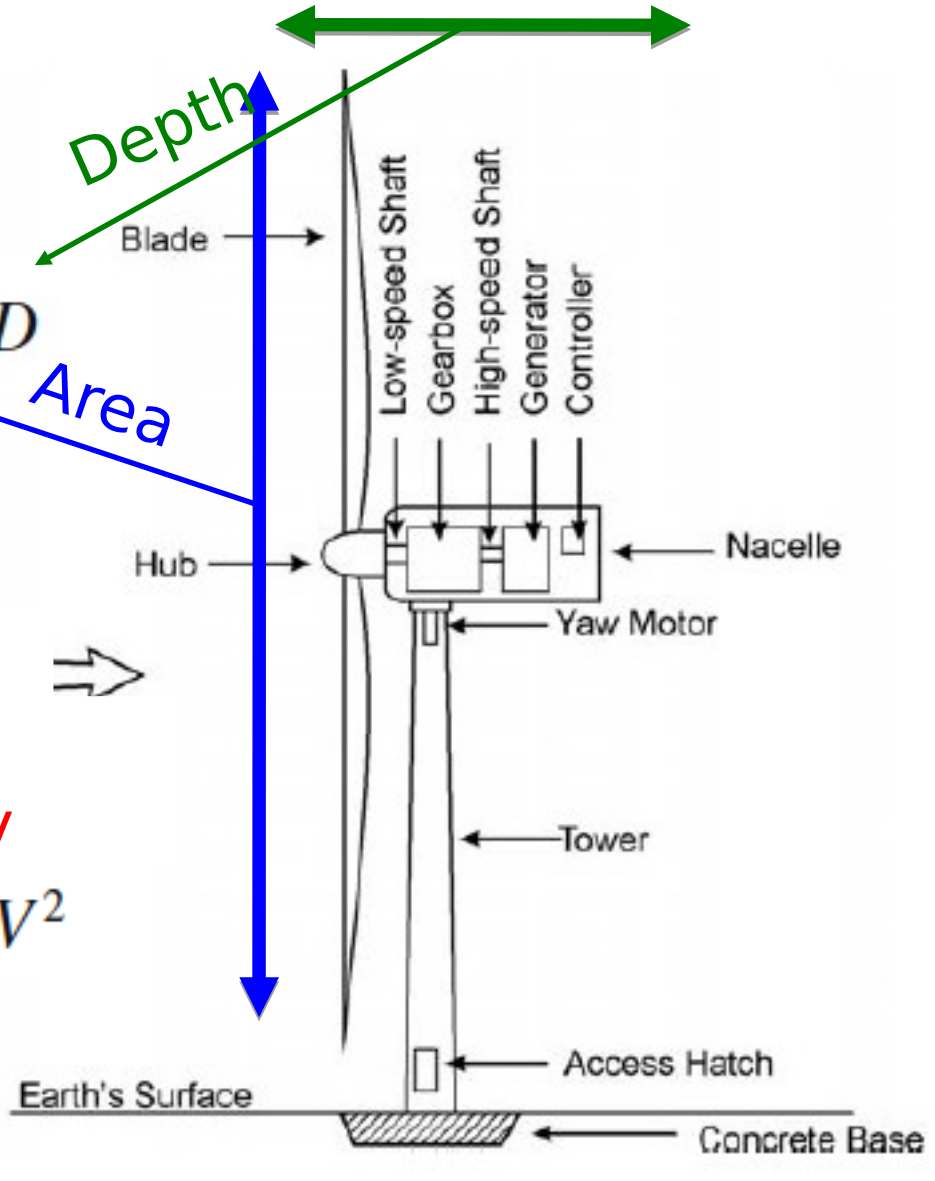
$$D = V \times T$$

Wind Velocity

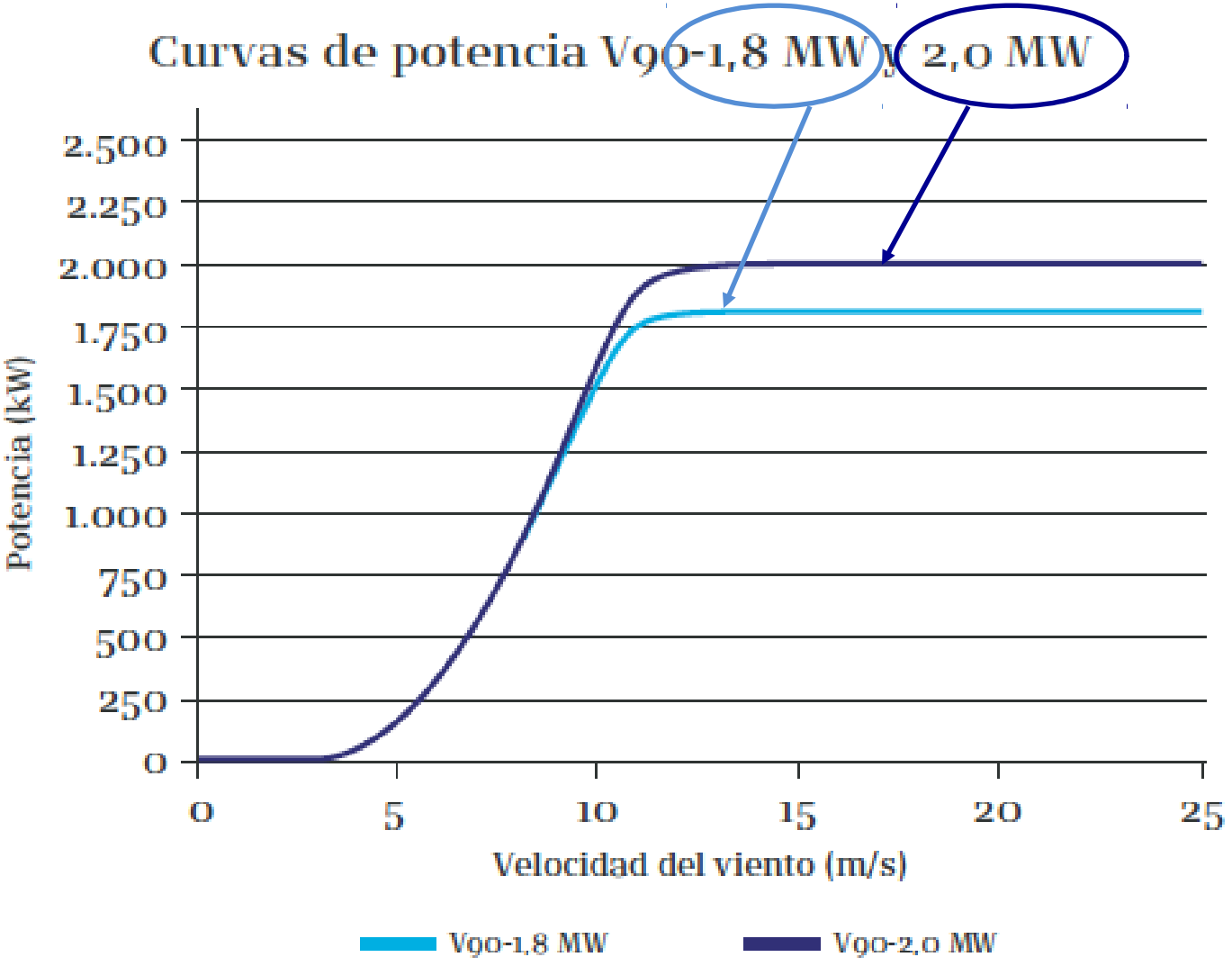
$$KE = \frac{1}{2} \times (\rho \times A \times V \times T) \times V^2$$

$$KE = \frac{1}{2} \times (\rho \times A \times V^3 \times T)$$

$$\text{Power} = (\frac{1}{2} \times \rho \times A \times V^3 \times T)/T$$



Curva teórica de potencia



Que Modelo Ajusta R (en la práctica) ?

$$E(Y|X_1, X_2, \ldots, X_p) = \mu_y + \sum_{m=1}^{M_o} \beta_m \phi_m (a_m^T \mathbf{x})$$

$\mu_y = E(Y)$

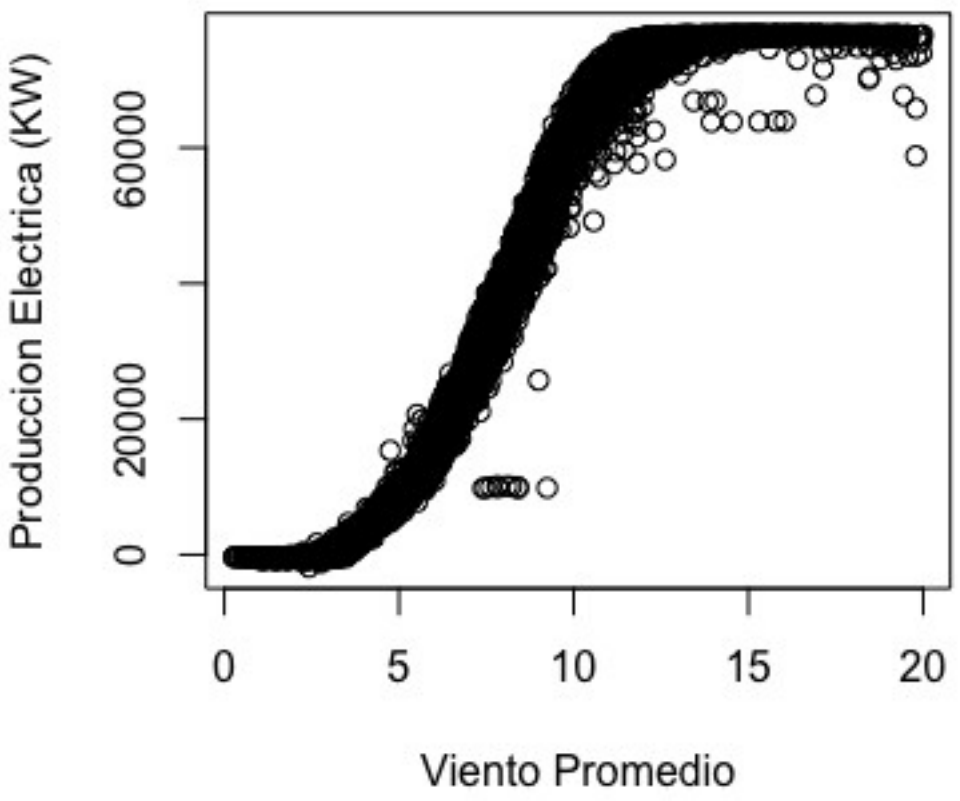
$\|a_m\| = 1$

$E(\phi_m(a_m^T x)) = 0$

$Var(\phi_m(a_m^T x)) = 1, m = 1, \ldots, M_o.$

Ajuste con R

prod	vw1	vw2	vw3
7315.500	5.7	5.5	5.5
5816.250	5.0	5.1	5.2
5233.317	5.1	4.1	5.3
2772.883	4.6	3.3	4.7
2780.933	3.6	3.1	3.3
1197.083	4.4	5.2	2.4
1664.917	4.8	5.1	2.8

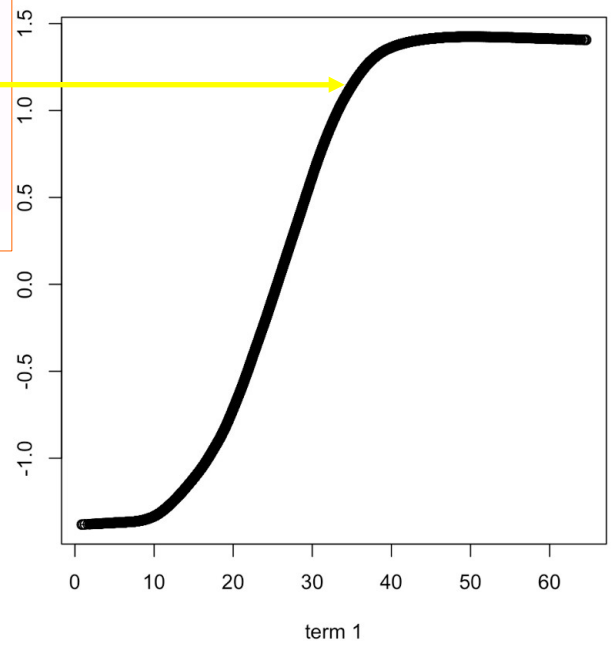



```
ajus.ppr <- ppr(prod ~ .,data=vtos, nterms = 2, max.terms = 2,trace=TRUE)
```

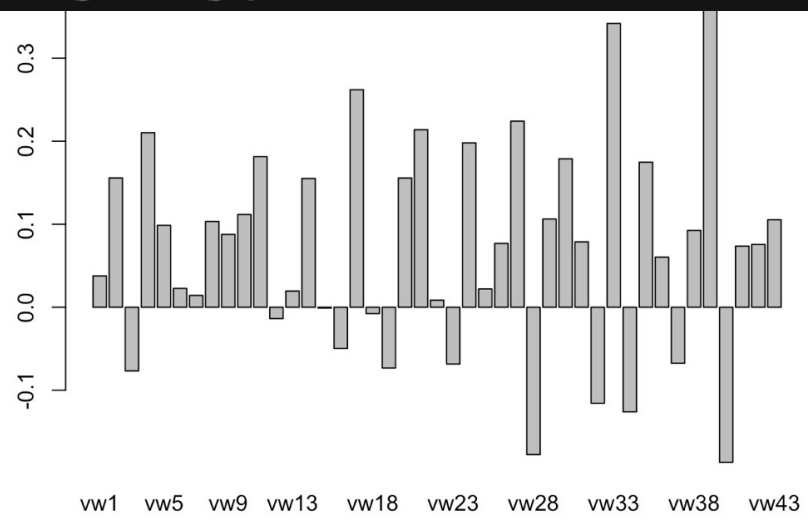
$$E(Y|X_1, X_2, \ldots, X_p) = \mu_y + \sum_{m=1}^{M_o} \beta_m \phi_m(a_m^T \mathbf{x})$$

```
Projection direction vectors:
      term 1         term 2
vw1    0.0377294292 -0.3148607207
vw2    0.1556777260  0.1447209286
vw3   -0.0767089594 -0.0023975299
vw4    0.2102003279  0.1906500940
vw40 -0.1867671914  0.1
vw41  0.0736125220  0.2733572416
vw42  0.0757829072 -0.0667818081
vw43  0.1054183205 -0.1567821373

Coefficients of ridge terms:
    term 1      term 2
27585.9312   563.0649
```




```
> c(ajus.ppr$yb,mean(prod,na.rm=T))
[1] 37279.11 37279.11
```

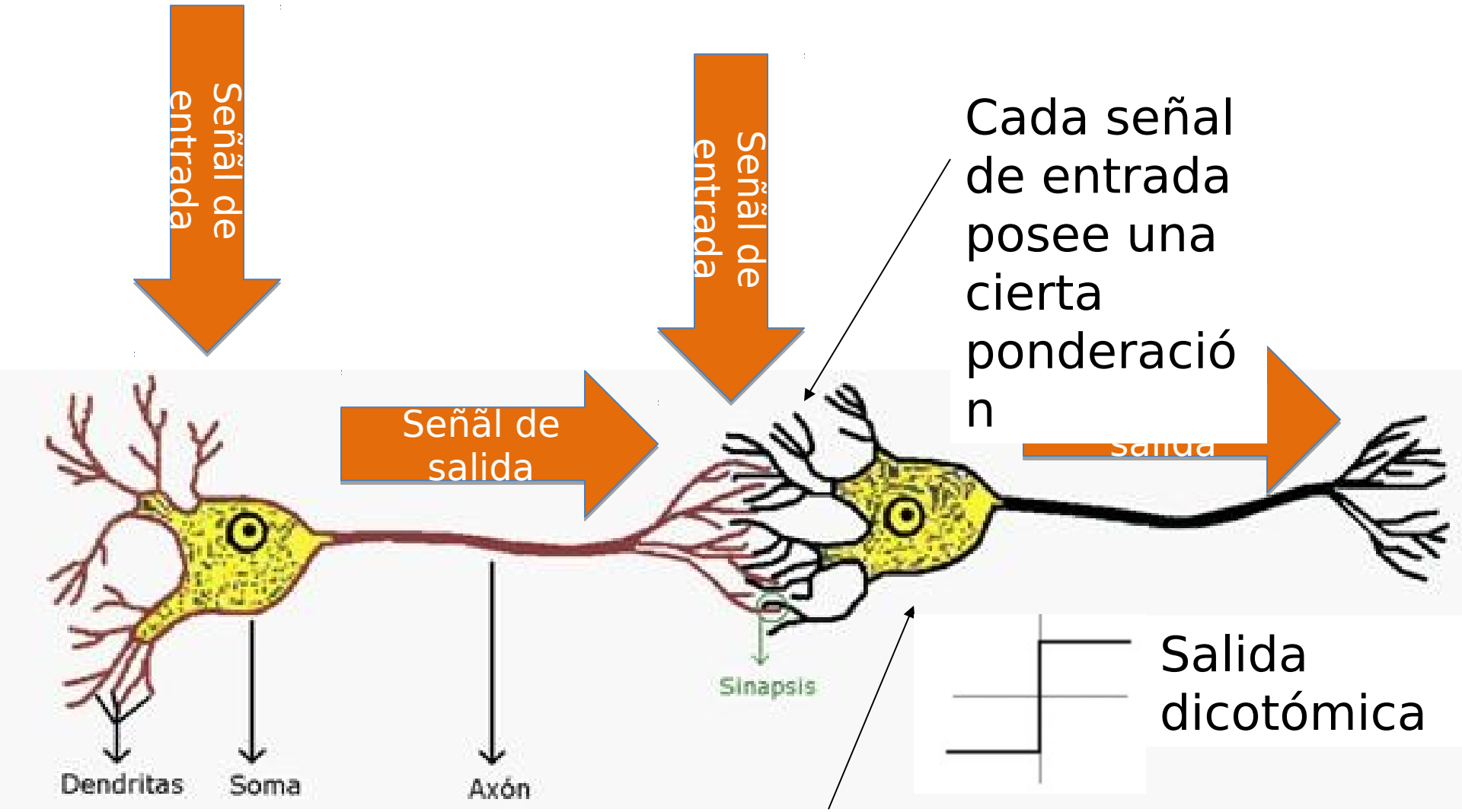


Redes Neuronales Artificiales

nnet

- Son métodos tipo “Black Box”
- Son buenos aproximadores de funciones
- Tendencia al sobreajuste (usar penalización)
- Sirven naturalmente tanto para Regresión como para Clasificación
- Particularmente útiles en el reconocimiento de patrones
- Se calibran mediante Gradient Descent Estocástico
- En el fondo, son modelos No Lineales encadenados
- Particularmente útiles cuando hay múltiples outputs

La Neurona

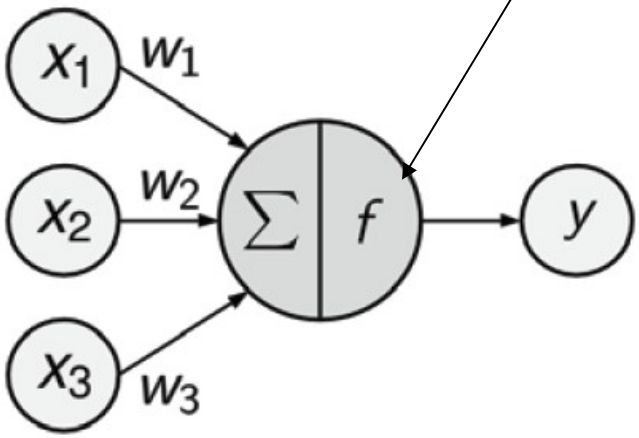


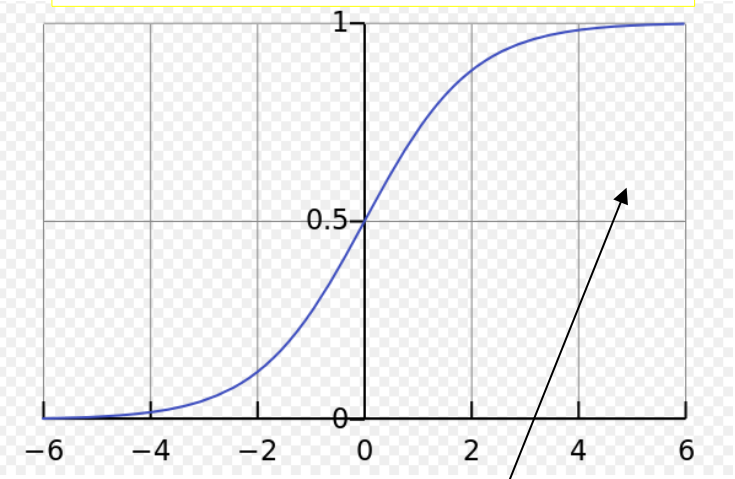
La neurona “suma” las señales ponderadas y “gatilla” una señal de

Elementos Básicos de una Red

Función de activación

$$f(x) = \frac{1}{1+e^{-(x)}}$$





$$f\left(\sum_{i=1}^{n} w_i x_i\right) = y(x)$$

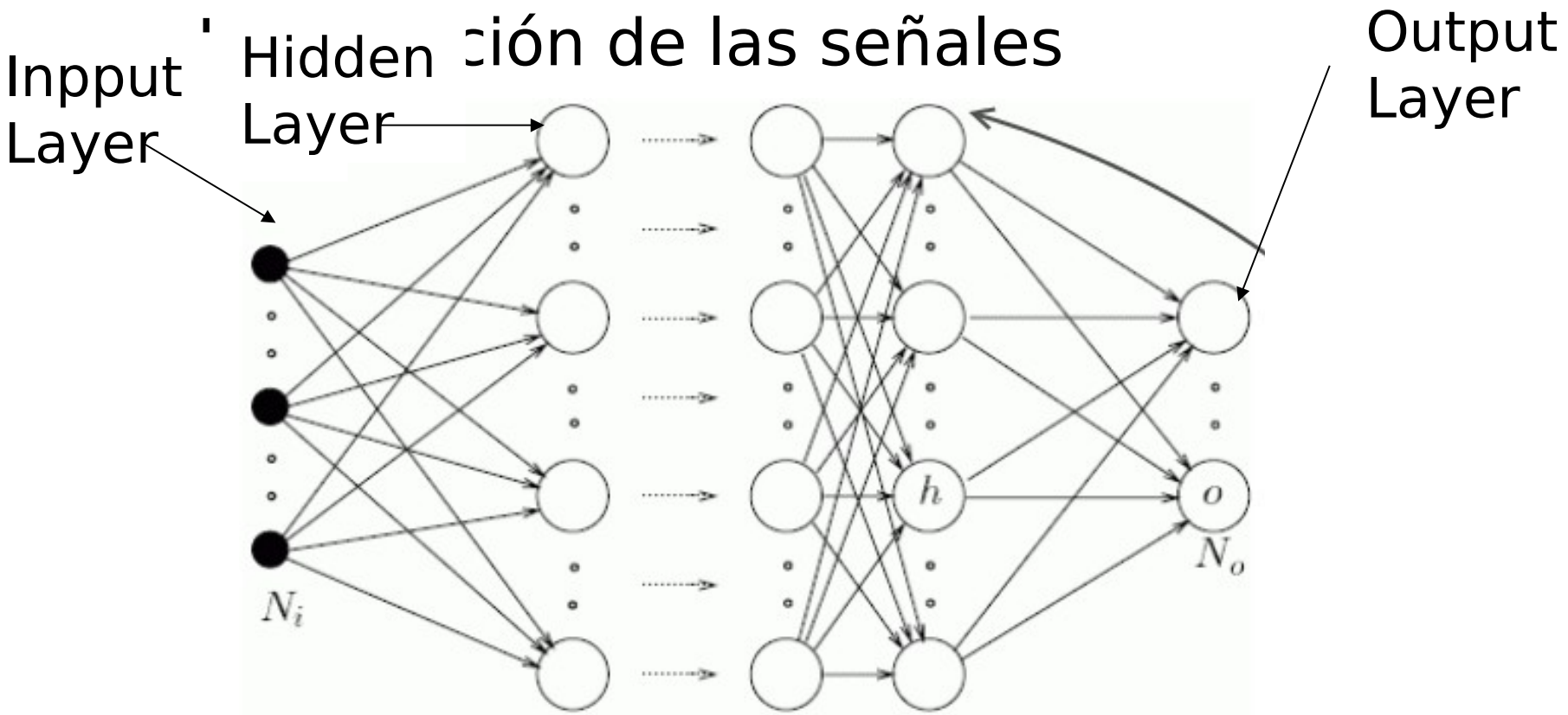
Función Sigmoidea o Logística

Tangente Hiperbólica

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

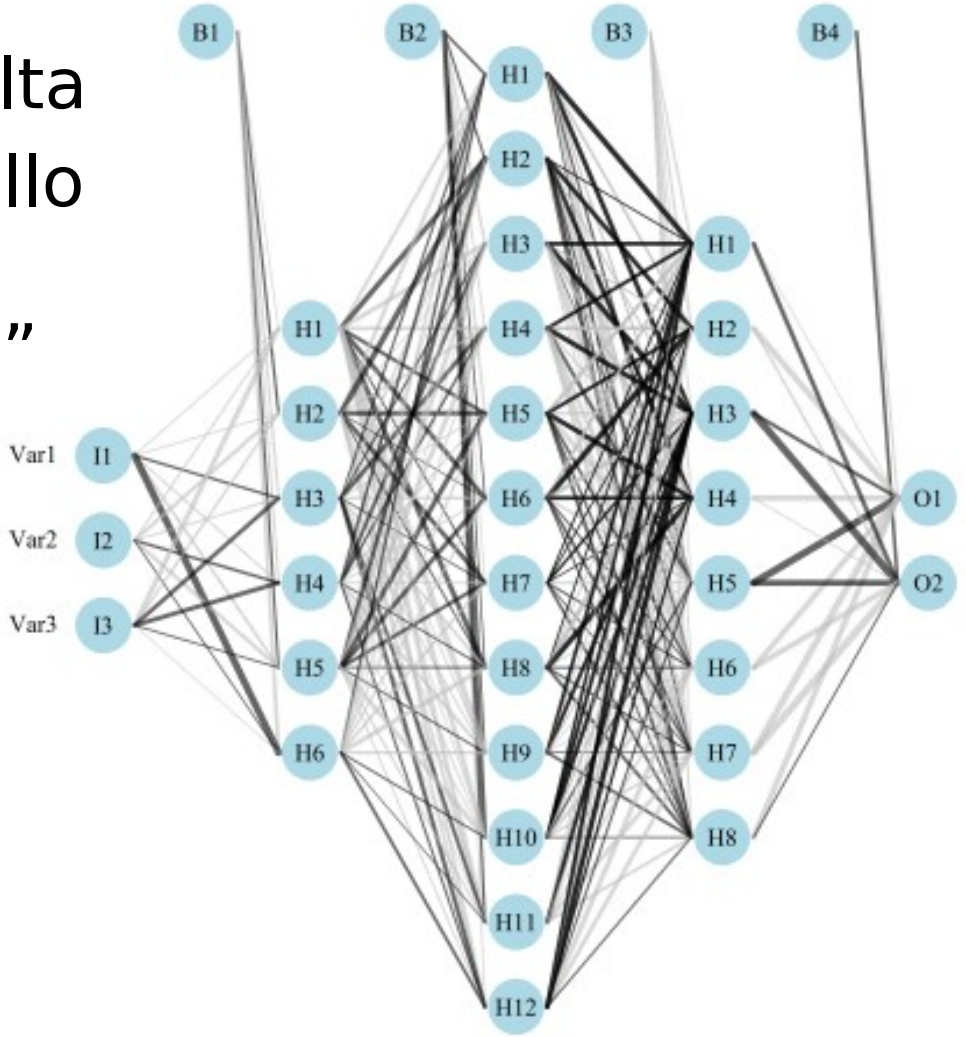
Topologías de Redes Neuronales

- La complejidad de una red depende de:
  - La cantidad de capas de nueronas
  - La cantidad de neuronas por capa
  - [illegible]ción de las señales

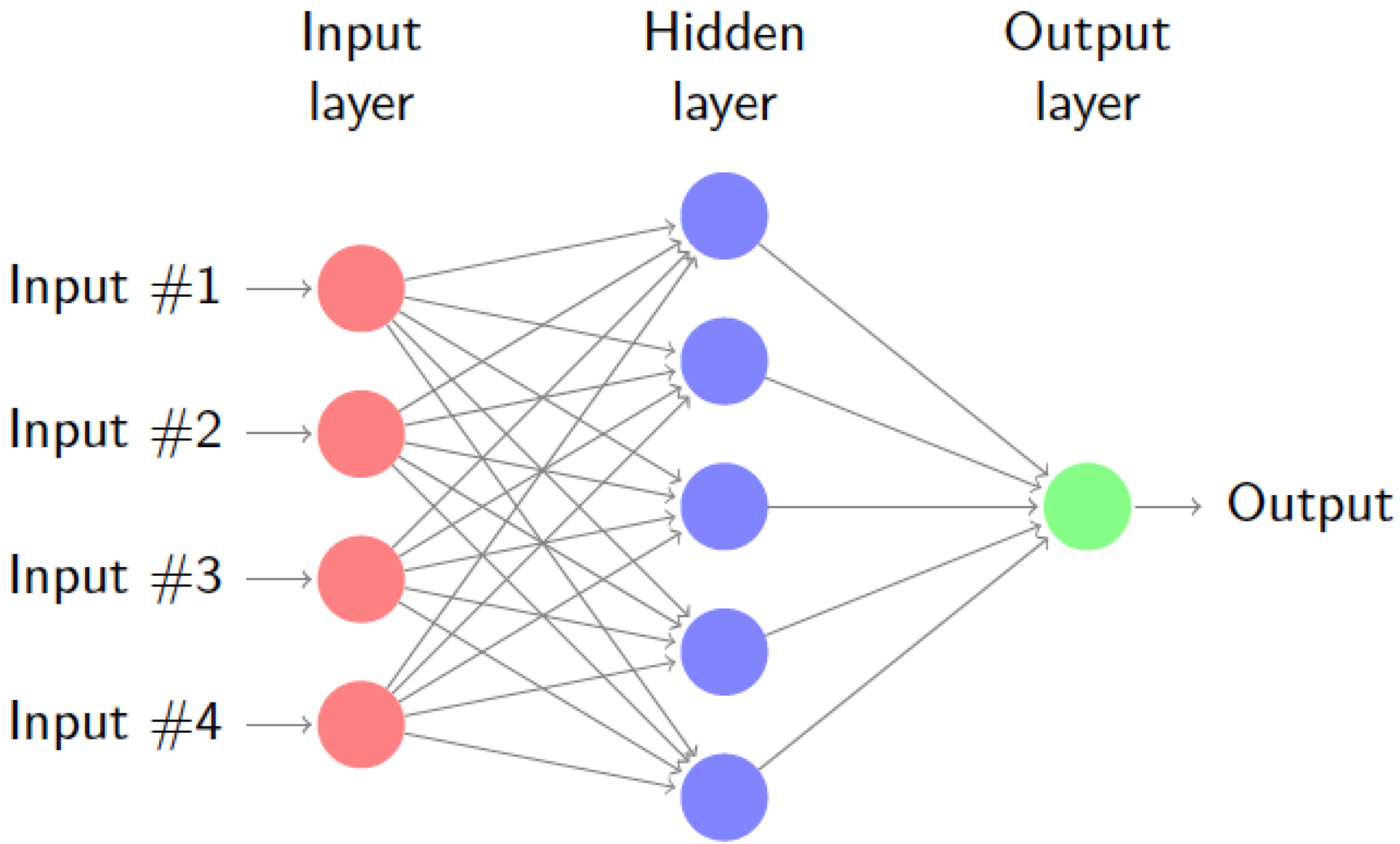


El Multilayer Perceptron

- Es la arquitectura mas difundida
- Al menos una capa oculta
- Es el modelo más sencillo (1 capa) que genera un “aproximador universal”
- Gran cantidad de parámetros a ser calibraods

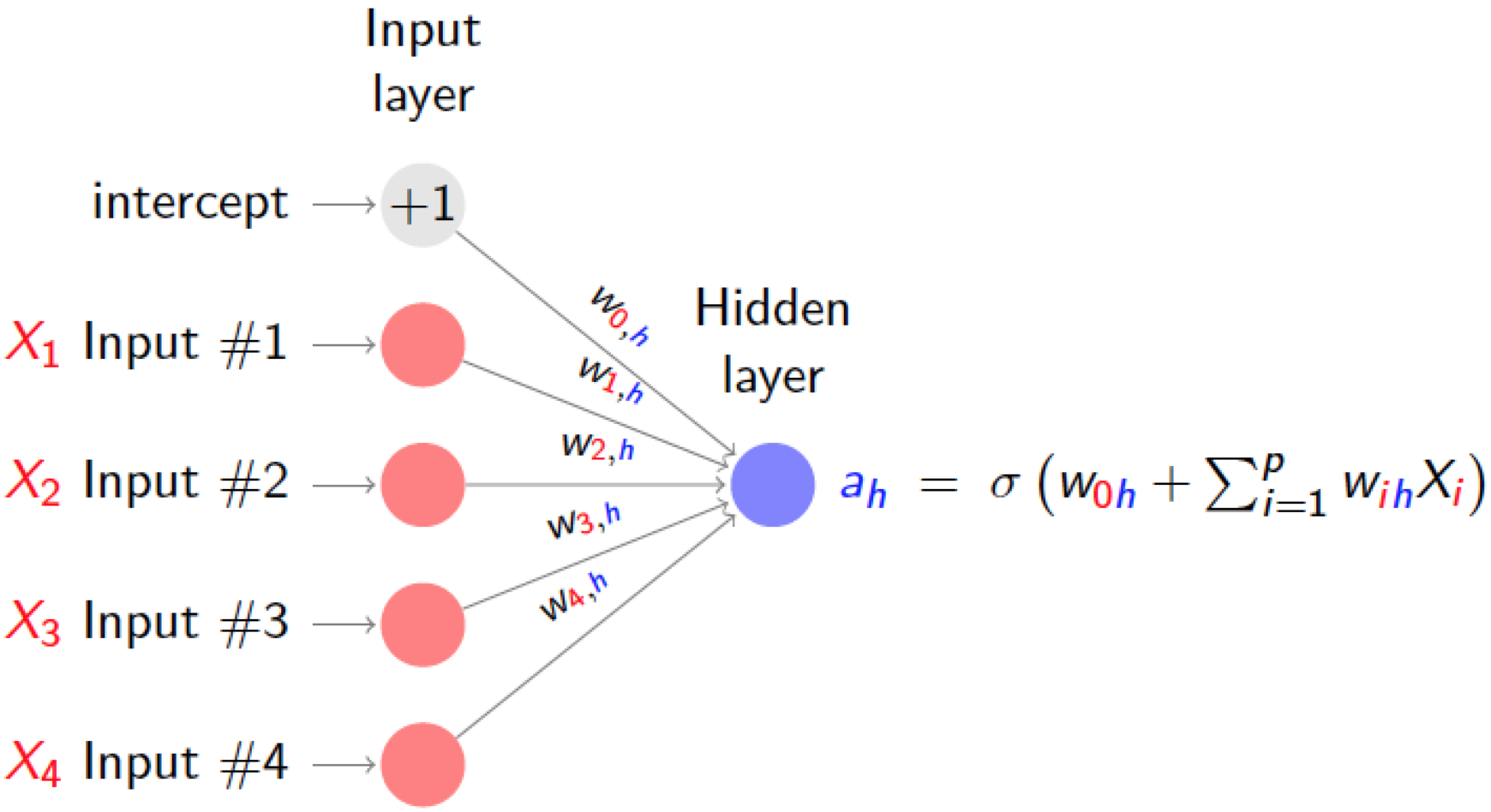


Redes Neuronales: Ejemplo



4 variables predictoras o inputs 5 unidades ocultas

Ejemplo de Red Neuronal



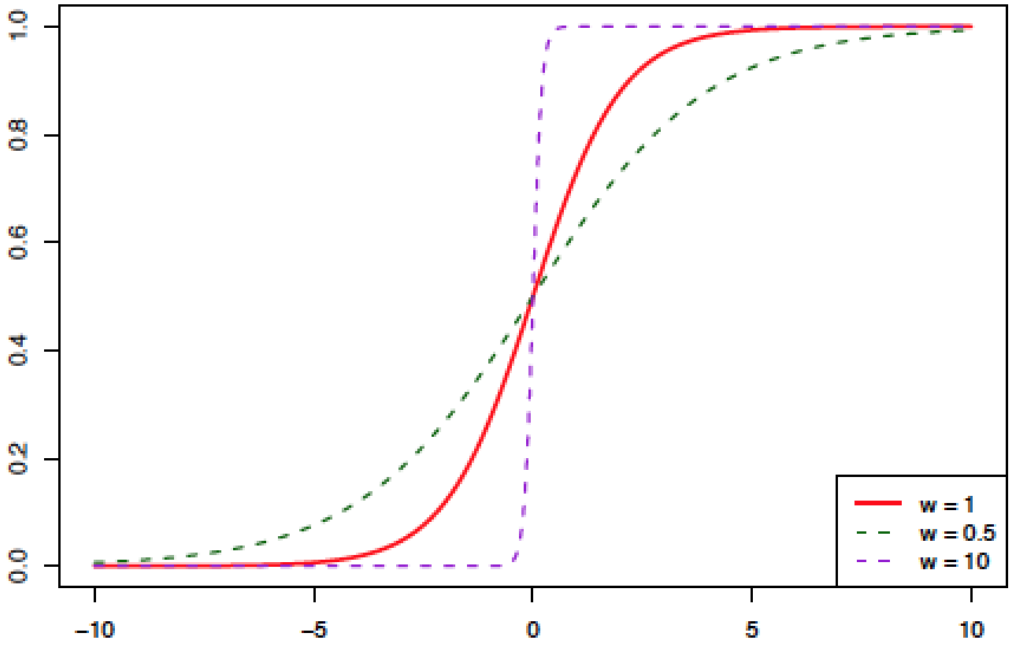
Red Neuronal

- Nodos: unidades de memoria o neuronas
- $w_{ih}^{(l)}$: son pesos (parámetros) l representa la layer, i indica el nodo del que sale, h el nodo al que llega
- El modelo de redes neuronales impone que cada nodo es una función no lineal de una combinación lineal de los nodos de la capa (o layer) anterior
- σ es una función no lineal, usualmente la sigmoidea

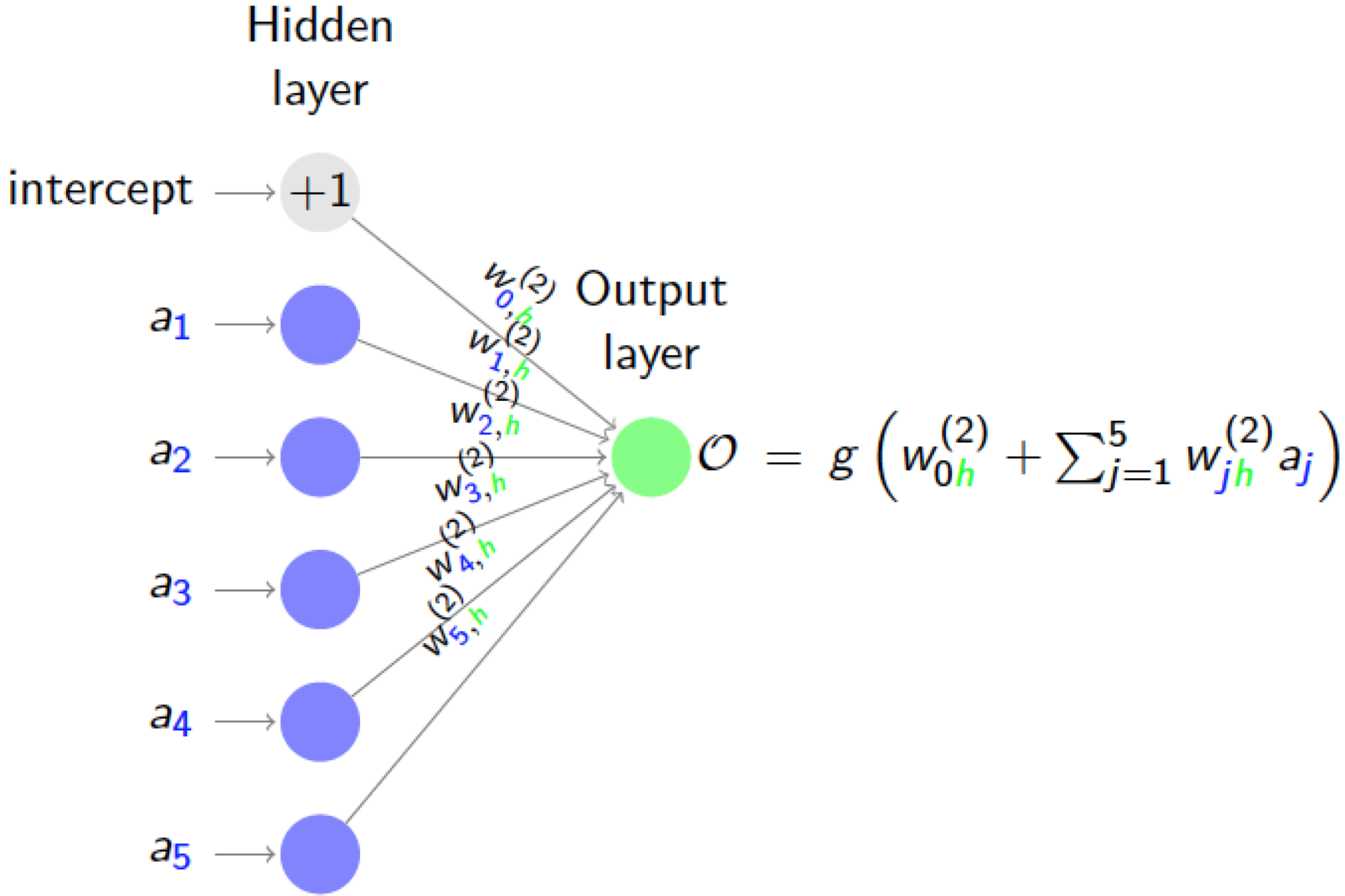
$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Graficamos

$$\sigma(wt) = \frac{1}{1 + e^{-wt}}$$



Red Neuronal: output



Red Neuronal: output

$$output = \mathcal{O} = g\left(w_{0h}^{(2)} + \sum_{j=1}^{5} w_{jh}^{(2)} a_j\right) \tag{2}$$

¿Cómo se elige la función g? Depende de la variable respuesta.

- En problemas de regresión (Y continua), g es la identidad
- Cuando la respuesta es binaria, g es la sigmoidea
- Cuando la respuesta es categórica, con K categorías, g es la identidad en cada nodo, pero el output final lleva una normalización:

$$\mathcal{O}_k = \widehat{p}\,(\text{categoría } k \mid (z_1, \ldots, z_K)) = \frac{e^{z_k}}{\sum_{h=1}^{K} e^{z_h}}$$

(transformación de la logística múltiple, o *función softmax*)
donde $z_k = w_{0k}^{(2)} + \sum_{j=1}^{5} w_{jk}^{(2)} a_j$.

Red Neuronal: relación con otros modelos

- Sin *hidden layers*, las redes neuronales son un modelo lineal generalizado
- (Vínculo con PPR): Las redes neuronales con una sola *hidden layer* tienen la misma forma que el modelo PPR. La diferencia es que PPR usa funciones no paramétricas y RN usa sigmoideas.
- Las RN representan una versatilidad de modelos, ya que pueden variar
  - cantidad de unidades en la capa oculta
  - cantidad de capas ocultas

Redes Neuronales: Ajuste

Todas las capas son funciones de las capas anteriores, que a su vez son funciones de las variables explicativas:

$$f(\mathbf{X}, \mathcal{W})$$

- $\mathcal{W}$ es la colección de pesos. ¿Cuántos? En el ejemplo $(p+1)H + (H+1) = 5 \cdot 5 + 6 = 31$ ¡muchos!
- Si contáramos con
  - una muestra de entrenamiento: $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$
  - una función objetivo: $L[Y, f(\mathbf{X}, \mathcal{W})]$
  - un algoritmo de optimización

  podríamos estimarla.
- Debemos penalizar la función objetivo en los pesos, para evitar el sobreajuste

Redes Neuronales: Ajuste

Queremos hallar los pesos $\mathcal{W}$ que resuelvan

$$min_{\mathcal{W}} \left\{ \frac{1}{n} \sum_{i=1}^{n} L\left[Y_i, f(\mathbf{X}_i, \mathcal{W})\right] + \lambda J(\mathcal{W}) \right\} \tag{3}$$

En realidad $J(\mathcal{W}) = \sum_{s=1}^{S} \lambda_s J_s(\mathcal{W})$, es el término de penalización

- Se usó primero la pérdida cuadrática,
  $J(\mathcal{W}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{l=1}^{L} \sum_{h=1}^{H_l} \left[w_{ih}^{(l)}\right]^2$ (*weight decay penalty*)
  (penalidad tipo Ridge)
- Luego penalidades de tipo Lasso (l_1), con el $|\cdot|$ en lugar del cuadrado (pesos mas ralos)
- Combinaciones de ambos (*elastic net*)

Redes Neuronales: Función objetivo

- Si la respuesta es continua, L es la pérdida cuadrática

$$L\left[Y_i, f(\mathbf{X}_i, \mathcal{W})\right] = \left[Y_i - f(\mathbf{X}_i, \mathcal{W})\right]^2$$

- Si la respuesta es categórica, L es la deviance binomial (multiplicada por (-1))

$$L\left[Y_i, f(\mathbf{X}_i, \mathcal{W})\right] = Y_{ik} log(f_k(\mathbf{X}_i, \mathcal{W}))$$

Redes Neuronales: Algoritmo

La implementación del ajuste está lleno de detalles y mejoras

- La función objetivo es convexa en f, pero no en los pesos, no es fácil encontrar óptimos, hallaremos óptimos locales
- f es una función diferenciable y L también lo es: usamos el método o búsqueda dirigida por el gradiente *(gradient descent)*.
- Se inicializa en valores aleatorios
- El gradiente se calcula mediante un algoritmo que se basa en la estructura jerárquica de los pesos: *back propagation*.
- El algoritmo se acelera combinándolo con selecciones aleatorias de los datos a actualizar: *(batch gradient descent)*.
- Los parámetros del método del gradiente son elegidos adaptivamente.

Ejemplo: Dígitos postales (ZIP code)

Problema de clasificación: buscamos un clasificador automático de dígitos manuscritos. Base de datos de 60.000 dígitos manuscritos de entrenamiento. Y otros 10.000 para testear. Por ejemplo:

Ejemplo: Dígitos postales (ZIP code)

Cada dígito está representado por una escala de grises de $28 \times 28 = 784$ píxeles, $(X_1, \ldots, X_{784})$. El valor que se guarda en cada píxel es un número positivo que indica la intensidad de gris presente en esa ubicación. Los 784 píxeles representan las covariables, la respuesta es un número de 0 a 9.
Presentamos la red ajustada por

- Efron, B. y Hastie T. (2016) *Computer Age Statistical Inference Algorithms, Evidence, and Data Science*. Cambridge University Press.
- Ajustado con el paquete `h20` de R.
- `MNIST` es la base de datos curada (LeCun y Cortes, 2010) disponible públicamente antes descripta.

Ejemplo: Dígitos postales (ZIP code), conclusión

Esa red, con los parámetros apropiadamente elegidos, da un error de clasificación del 0,93 % en el conjunto oficial de testeo. Random forests 2,8 % de error, modelo lineal generalizado 7,2 % Acá los 93 dígitos mal clasificados (verdadero en azul, clasificación en rojo)

Ejemplo: Dígitos postales (ZIP code), evolución

Para los datos `MNIST`, la mejor red neuronal daba

- En el 2008 un error del 1,6 %
- En el 2016 un error del 0,93 %

En realidad, el error estándar de la tarea de clasificación de un conjunto parecidos es mayor, ya que los datos de testeo han sido implícitamente usado por los diversos métodos para "tunear" los procedimientos.

Entrenamiento de una Red Nuronal: BackPropagation Algorithm

- Basado en Gradient Descent Estocástico (dato por dato)
- Busca el mínimo del Error en el espacio de los pesos.
- Combate la sobre-parametrización mediante lo “Estocástico”.
- 0) Se inicializan los pesos
- 1) Se ingresa un vector de features
- 2) Se calcula la salida (basada en los pesos)
- 3) Se calcula el error
- 4) Se calcula el graiente del error de atras para adelante (Back-Propagation)
- 5) Se ajustan los pesos mediante el gradiente