Aprendizaje Profundo (Deep Learning)

Redes Neuronales

César Gómez

2 de febrero de 2022

Perceptron

Una red neuronal toma un vector de p variables $X = (X_1, \ldots, X_p)$ y construye una función no lineal $f(X)$ para predecir una respuesta Y.

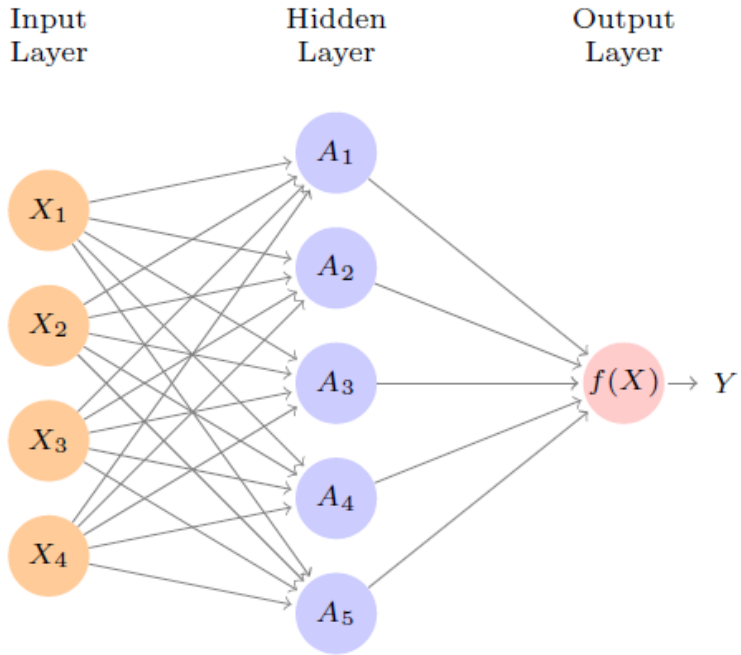

Figura 1

En la figura, 1 Se ilustra una red neuronal simple *de una capa*, *feed-forward*, también denominada *perceptrón de una capa*. Este modelo predice una respuesta cuantitativa con base en 4 predictores $X_1, \ldots, X_4$ que constutuyen las *unidades* en la *capa* de entrada.

- Las flechas indican que cada una de las unidades de entrada alimenta (*feed*) cada una de las K unidades ocultas $A_1, \ldots, A_K$.
- Esta red neuronal, considerada como un modelo, posee la forma

$$\begin{aligned} f(X) &= \beta_0 + \sum_{k=1}^{K} \beta_k h_k(X), \\ &= \beta_0 + \sum_{k=1}^{K} \beta_k g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j). \end{aligned} \tag{1}$$

- Cada una de las K unidades $A_1, \ldots, A_K$ de la *capa oculta* biene dada por

$$A_k = h_k(X) = g(w_{k0} + \sum_{j=1}^{p} w_{kj} X_j) \tag{2}$$

- Todos los parámetros $\beta_0, \ldots, \beta_K$ y $w_{10}, w_{11}, \ldots, w_{Kp}$ deben ser estimados a partir de los datos.
- $g(\cdot)$ es una función denominada función de activación.

De uso frecuente se encuentran: La actvación *sigmoide*

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}. \tag{3}$$

Y la función de activación RELU (*Rectified Linear Unit*)

$$g(z) = (z)_+ = \begin{cases} 0 & \text{si } z < 0 \\ z & \text{c.o.c} \end{cases} \tag{4}$$

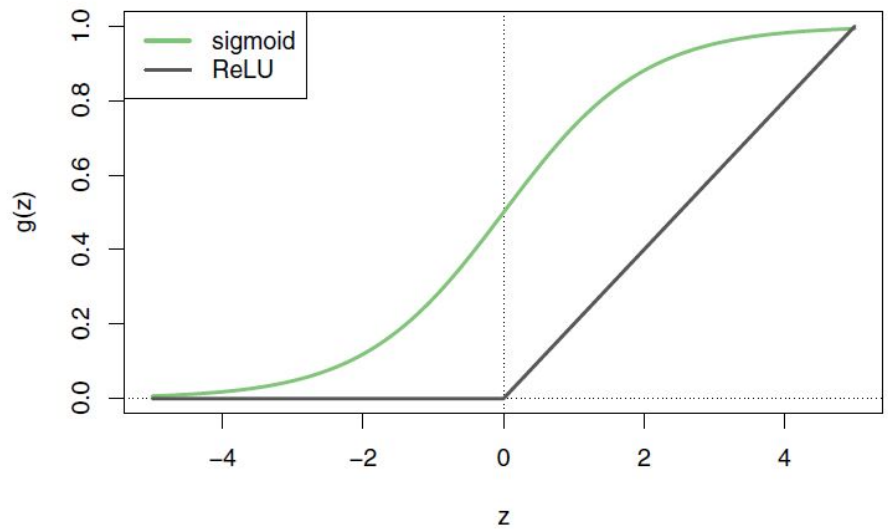

Figura 2

- La no lienealidad de la activación $g(\cdot)$ es esencial. De otra forma el modelo en (1) para $f(X)$ terminaría constituyendo un modelo lineal.
- Considere un ejemplo simple con $p = 2$ predictores $X = (X_1, X_2)$ y 2 unidades en la capa oculta $h_1(X)$, $h_2(X)$, con $g(z) = z^2$. Los demás parámetros se especifican como sigue:

$$\begin{aligned} \beta_0 = 0, \quad \beta_1 = \tfrac{1}{4}, \quad \beta_2 = -\tfrac{1}{4}, \\ w_{10} = 0, \quad w_{11} = 1, \quad w_{12} = 1, \\ w_{20} = 0, \quad w_{21} = 1, \quad w_{22} = -1, \end{aligned} \tag{5}$$

Con estos valores de los parámetros se tiene que

$$\begin{aligned} h_1(X) &= (0 + X_1 + X_2)^2, \\ h_2(X) &= (0 + X_1 - X_2)^2, \end{aligned} \tag{6}$$

Y para el modelo $f(X)$ tenemos

$$\begin{aligned} f(X) &= 0 + \frac{1}{4}(0 + X_1 + X_2)^2 - \frac{1}{4}(0 + X_1 - X_2)^2 \\ &= \frac{1}{4}\left[(X_1 + X_2)^2 - (X_1 - X_2)^2\right], \\ &= X_1 X_2. \end{aligned} \tag{7}$$

Ajuste de una red neuronal

El ajuste de una red neuronal requiere estimar los parámetros desconocidos. Para una respuesta cuantitativa, normalmente se utiliza la pérdida de error cuadrático, por lo que los parámetros se estiman de tal forma que minimicen

$$\sum_{i=1}^{n}(y_i - f(x_i))^2. \tag{8}$$

Perceptron Multicapa

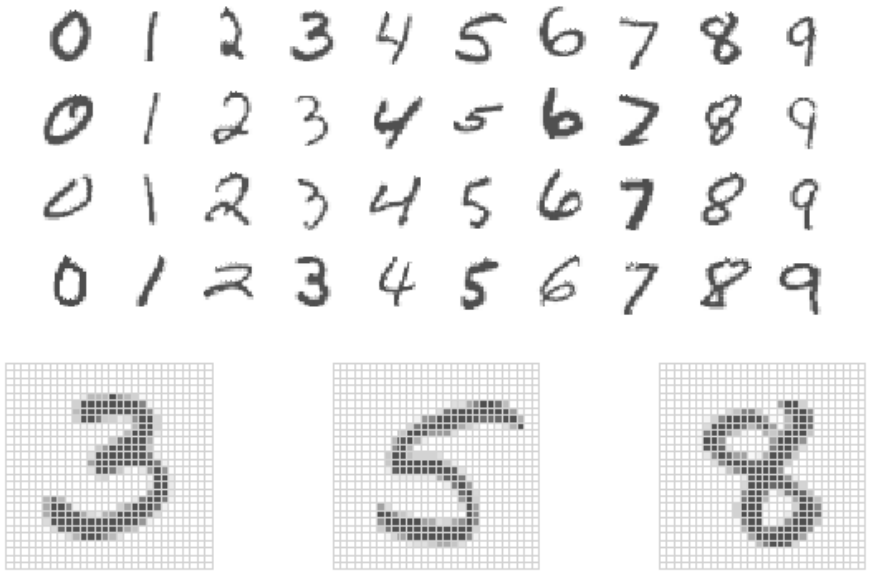

Figura 3: Ejemplos de dígitos manuscritos del corpus MNIST. Cada imagen en escala de grises posee 28x28=784 píxeles, cada uno de los cuales es un número de 8 bits (0-255) que representa que tan oscuro es ese píxel.

Perceptron Multicapa

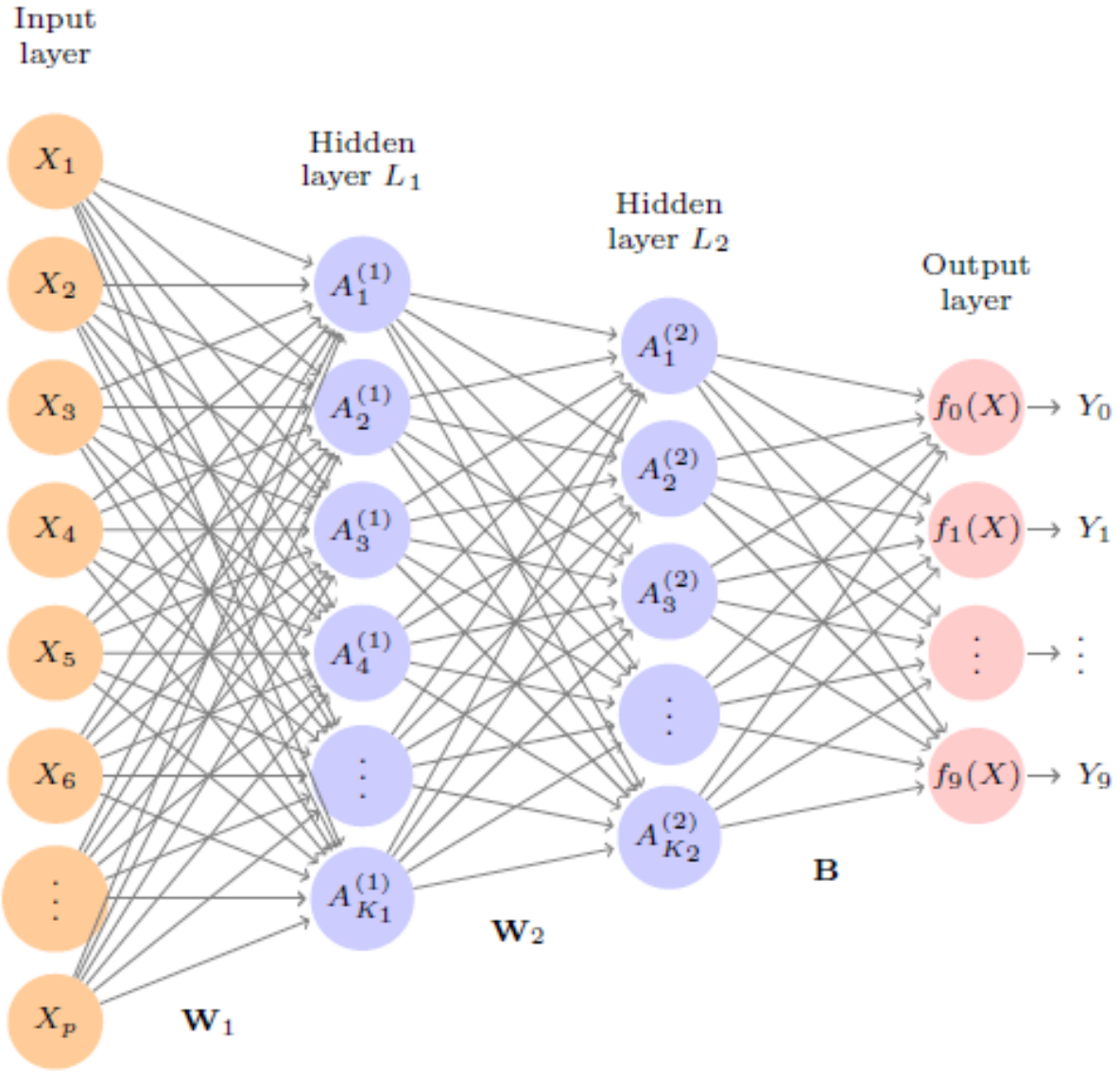

Figura 4

- En la figura (4) se ilustra el diagrama de una red neuronal con 2 capas ocultas adecuada para el problema de dígitos manuscritos MNIST. La capa de entrada posee 784 unidades. Las 2 capas ocultas poseen $K_1 = 256$ y $K_2 = 128$ unidades respectivamente, la capa de salida posee 10 unidades. Junto con interceptos (que se denominan unidades de sesgo 'biases'), esta red posee 235.146 parámetros.
- La red posee 10 unidades de salida que realmente representan una sola variable cualitativa (números 0-9) y por lo tanto las variables correspondientes están bien correlacionadas.

- Para la primera capa oculta, se tiene

$$\begin{aligned} A_k^{(1)} &= h_k^{(1)}(X) \\ &= g(w_{k0}^{(1)} + \sum_{j=1}^{p} w_{kj}^{(1)} X_j); \quad k = 1, 2, \ldots K_1. \end{aligned} \tag{9}$$

- Para la segunda capa se tiene

$$\begin{aligned} A_l^{(2)} &= h_l^{(2)}(X) \\ &= g(w_{l0}^{(2)} + \sum_{j=1}^{p} w_{lj}^{(2)} A_j^{(1)}); \quad l = 1, 2, \ldots K_2. \end{aligned} \tag{10}$$

Capa de saliada

$$Z_m = \beta_{m0} + \sum_{l=1}^{K_2} \beta_{ml} A_l^{(2)}; \quad m = 0, \ldots, 9. \tag{11}$$

Finalmente las probabilidades condicionales de la clase dado el vector de atributos, la obtenemos por medio de la función de activación *softmax*

$$f_m(X) = \Pr(Y = m|X) = \frac{e^{Z_m}}{\sum_{l=0}^{9} e^{Z_l}}. \tag{12}$$

Para entrenar esta red, colo la respuesta es cualitativa se buscan estimaciones de los parámetros que minimicen el negativo de la log-verosimilitud

$$-\sum_{i=1}^{n} \sum_{m=0}^{9} y_i \log(f_m(x_i)), \tag{13}$$

- Sumando el número de coeficientes en $\boldsymbol{W}^{(1)}$, $\boldsymbol{W}^{(2)}$ y $\boldsymbol{B}$ se obtienen 235.146 parámetros, con todo, más de $785 \times 9 = 7065$ parámetros que se necesitan con una regresión logística multinomial.
- Recuerdese que hay 60000 imágenes en el conjunto de entrenamiento. Aunque este paresca un conjunto de entrenamiento grande, hay casi 4 veces más parámetros en el modelo por lo que algún tipo de regularización es necesario para mitigar el sobre ajuste.

Método	Error de prueba
Red Neuronal + Regularización Ridge	2.3 %
Red Neuronal + Regularización *Dropout*	1.8 %
Regresión logística multinomial	7.2 %
Análisis de discriminante lineal	12.7 %

Cuadro 1: Tasas de error de varios métodos en la base de datos MNIST.

Sobre el ajuste o entrenamiento de RNN

En el caso del perceptron simple, por ejemplo, dadas observaciones (x_i, y_i), $i = 1, \ldots, n$ el modelo se puede ajustar, resolviendo el problema de mínimos cuadrádos no-lineales

$$\underset{\{w_k, \beta_k\}_0^K}{\text{minimize}} = \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2, \tag{14}$$

donde

$$f(x_i) = \beta_0 + \sum_{k=1}^{K} \beta_k g \left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{ij} \right). \tag{15}$$

Desenso de gradiente- Gradient descent

Suponga que se representan todos los parámetros en un vector θ, entoneces la función objetivo se puede representar como

$$R(\theta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2, \tag{16}$$

La idea del '*gradient descent*' es muy simple

1. Comience con un valor inicial θ^0 para todos los parámetros en θ, y haga $t = 0$.
2. Itere hasta que el objetivo falle en decrecer su valor
   - Encuentre un vector δ que ejerza un pequeño cambio en θ, tal que $\theta^{t+1} = \theta^t + \delta$ *reduce* el objetivo. O sea $R(\theta^{t+1}) < R(\theta^t)$
   - Haga $t \leftarrow t + 1$.

Backpropagation - Retro propagación

Como encontrar direcciones en las cuales mover a θ de manera que disminuye el el objetivo $R(\theta)$?.

- El *gradiente* de $R(\theta)$ es evaluado en valor actualizado $\theta = \theta^m$

$$\nabla R(\theta^m) = \left.\frac{\partial R(\theta)}{\partial \theta}\right|_{\theta=\theta^m}. \tag{17}$$

y se considera el siguiente exquema de actualización para θ

$$\theta^m \leftarrow \theta^m - \rho \nabla R(\theta^m). \tag{18}$$

- Debido a que $R(\theta) = \frac{1}{2}\sum_{1=1}^{n}(y_i - f_\theta(x_i))^2$ es una suma, el gradiente también es una suma sobre n observaciones. Por eso nos concentramos en 1 de los términos

$$R(\theta)_i = \frac{1}{2}\left(y_i - \beta_0 + \sum_{k=1}^{K}\beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj}x_{ij}\right)\right)^2. \tag{19}$$

Backpropagation - programación dinámica

$$\frac{\partial L}{\partial w(h_{r-1}, h_r)} = \frac{\partial L}{\partial o} \cdot \underbrace{\left[\sum_{[h_r, h_{r+1}, \ldots, h_{r_k}, o] \in \mathcal{P}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{h_i} \right]}_{\text{backpropagation calcula } \Delta(h_r, o) = \frac{\partial L}{\partial h_r}} \frac{\partial h_r}{\partial w(h_{r-1}, h_r)} \tag{20}$$

Regularización I. - Ridge - Lasso.

La red en (4) posee al rededor de 235000 pesos (parámetros), lo cual hace propenso el modelo a sobreajustarse. La primera fila de la tabla [1] usa regularización *Ridge* que se consigue añadiendo a la función de perdida un término de regularización

$$R(\theta, \lambda) = -\sum_{i=1}^{n}\sum_{m=0}^{9} y_{im}\log(f_m(x_i)) + \lambda \sum_{j} \theta_j^2. \qquad (21)$$

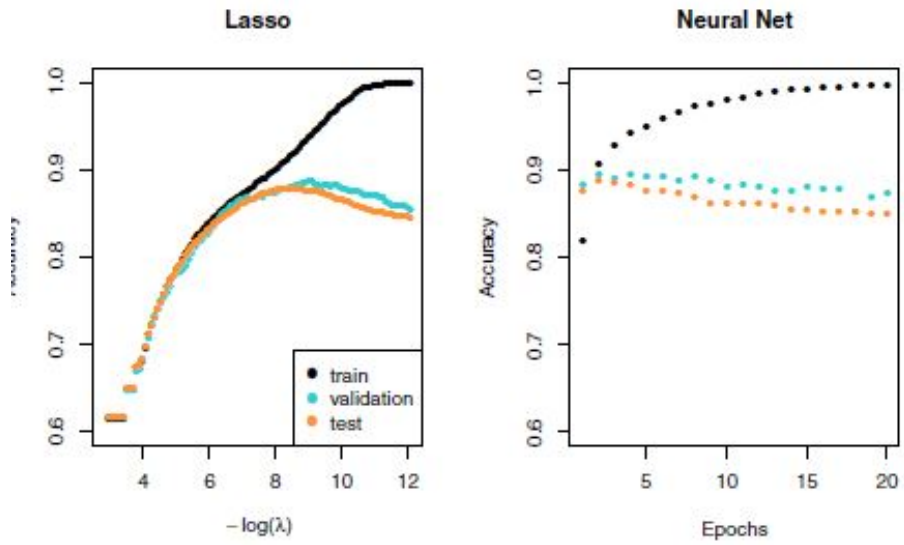

Figura 5: Precisión de una perdida regularizada con Lasso y una red neuronal sobre el conjunto `IMDb`, en el Lasso, el eje x ilustra $-\log(\lambda)$, mientras que para la red neuronal ilustra 'epocas' (*el número de veces que el algoritmo de ajuste recorre todo el conjunto de entrenamiento*). Ambos presentan una tendencia al sobre ajuste y aproximadamente la misma precisión de prueba.

Regularización II - Terminación Temprana (*Early Stoping*)

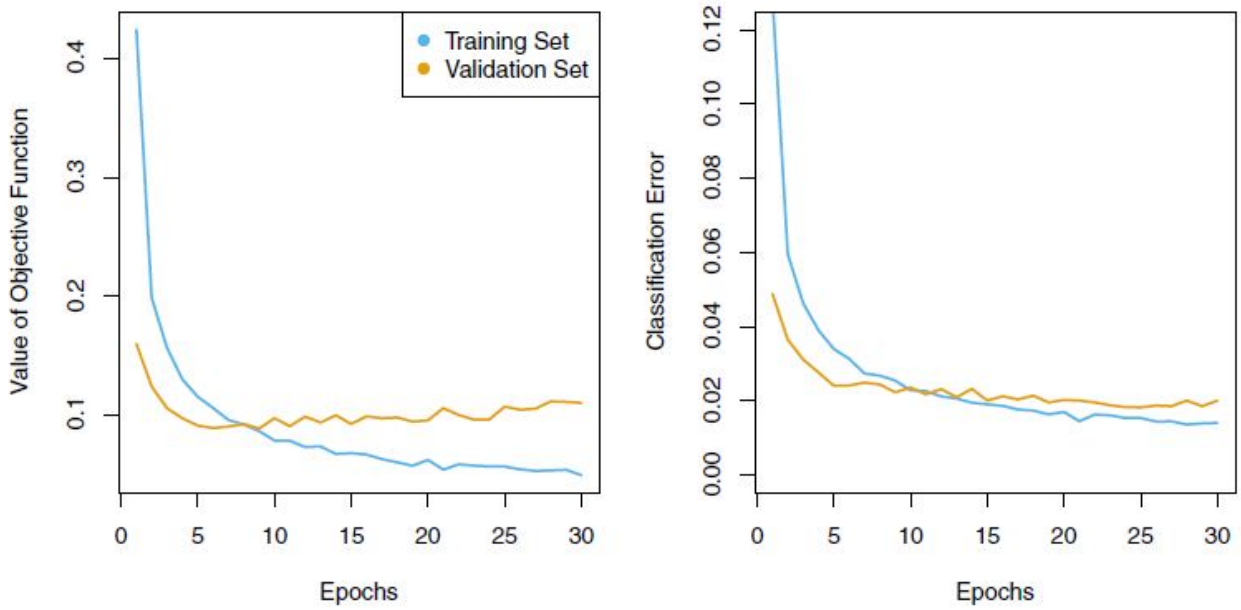

Figura 6: Evolución de errores de validación y prueba para el perceptron multicapa (4) sobre los datos MNIST, como una función de las épocas de entrenamiento

Regularización III *Dropout*

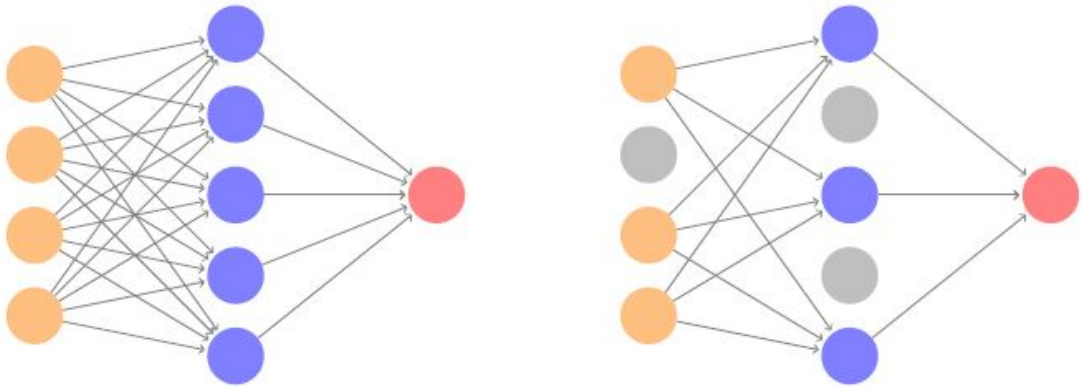

Figura 7: Aprendizaje con *dropout*: En la izquierda una red totalmente conectada, en la *derecha*, una red con *dropout* en la capa de entrada y la capa oculta. Los nodos en gris constituyen una **fracción** del total de nodos por capa, seleccionada aleatoriamente e ignorada en una instancia (época) de entrenamiento.

Redes convoluciónales o de convolución. CNN

(*Convolutional neural network*)

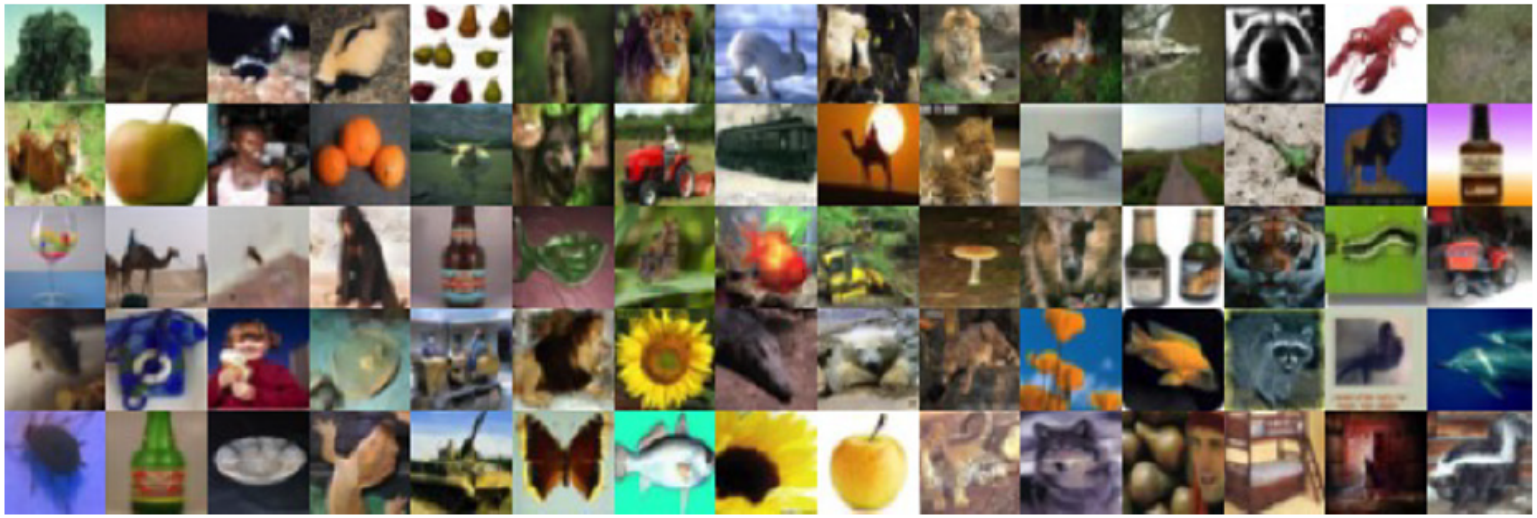

Figura 8: Una muestra de imágenes de la base de datos CIFAR100.

La base de datos CIFAR100 consiste de 60000 imágenes etiquetadas de acurdo a 20 superclases (p.e mamiferos marinos), con 5 clase por superclase (p.e delfin,castor,foca, ballena,nutria). Cada imagen posee una resolución de 32x32 pixeles, con 3 números de 8 bits que representan rojo,verde y azul.

- Cada una de las imágenes está organizada en un arreglo 3-dimensional denominado *feature map* (mapa de caraterísticas). Donde los primeros 2 ejes son espaciales y el tercero es el *canal* que representa cada uno de los 3 colores.

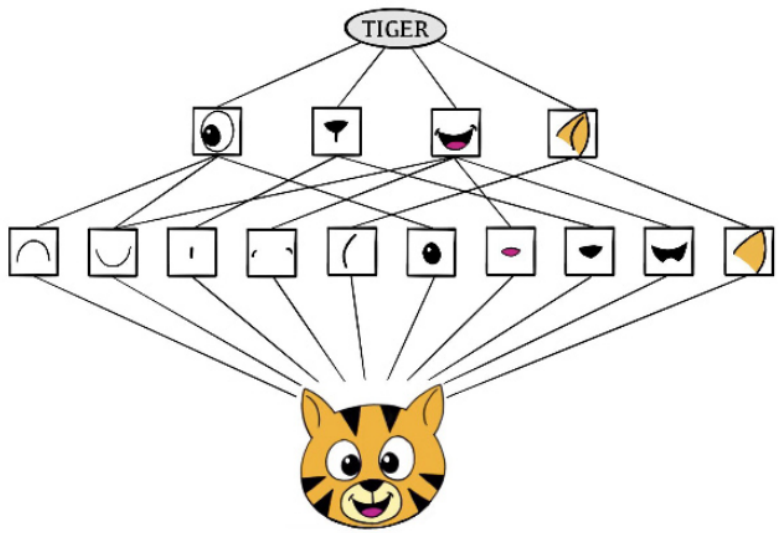

Figura 9: Esquemaqtización del trabajo de clacificación de una red convolucional.

Capas de convolución y capas *max pooling*

Una red convolutiba lleva a cabo su operación principalmente por medio de la combinación de 2 capas denominadas respectivamente capas de convolución y capas de pooling.

- Las capas de convolución detectan instancias de patrones pequeños en la imagen como bordes, rectas, etc. Que en capas más profundas de la red son luego codificados posiblemente como ojos, orejas, etc.
- Una capa de convolución está conformada por un grán número de *filtros de convolución* que se puede interpretar como una plantilla que detecta la presencia de una caracteristica de manera local en una imagen.

Filtros de convolución

Para comprender la acción de un filtro de convolución, considerese un ejemplo simple de una imagen 4x3

$$\text{Imagen original} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}, \quad \text{Filtro de convolución} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

$$\text{'convolved image'} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Filtros de convolución

- La idea es que si una submatriz 2x2 de la imagen original, luce similar(presenta un patrón) al filtro en tonces se obtendrá un valor alto en la imagen 'resultante' (convolve image) y viceversa, se obtendrá un valor pequeño cuando no hay semejanza entre la submatrix 2x2 de la imagen y el filtro.
- Por lo tanto la imagen resultante (producto de la convolución) resalta regiones donde la imagen se asemeja al filtro de convolución.
- La siguiente figura iulustra la acción de los filtros de convolución sobre una imagen de 192x179 de un tigre. Cada filtro de convolución es una imagen 15x15 compuenta en principio de ceros a exepción de una banda central de 1s en un caso vertical y en otro caso horizontal.

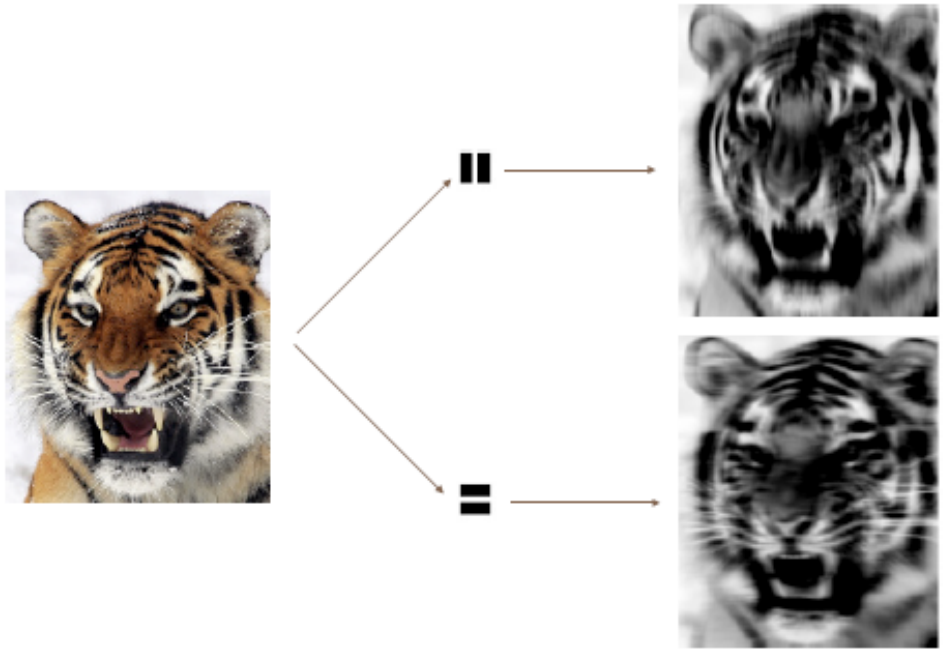

Figura 10: .

Capas de pooling

- Una capa de *pooling* proporciona una forma de condensar una imagen grande en una más pequeña. Hay muchas formas posibles de desarrollar un pooling, la operración de *max pooling* resume cada bloque 2x2 (sin intersecciones) en un único número.
- Esto reduce cada dimensión por un factor de 2 a la vez que proporciona cierta invarianza local.

$$\text{max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

Arquitectura de una CNN

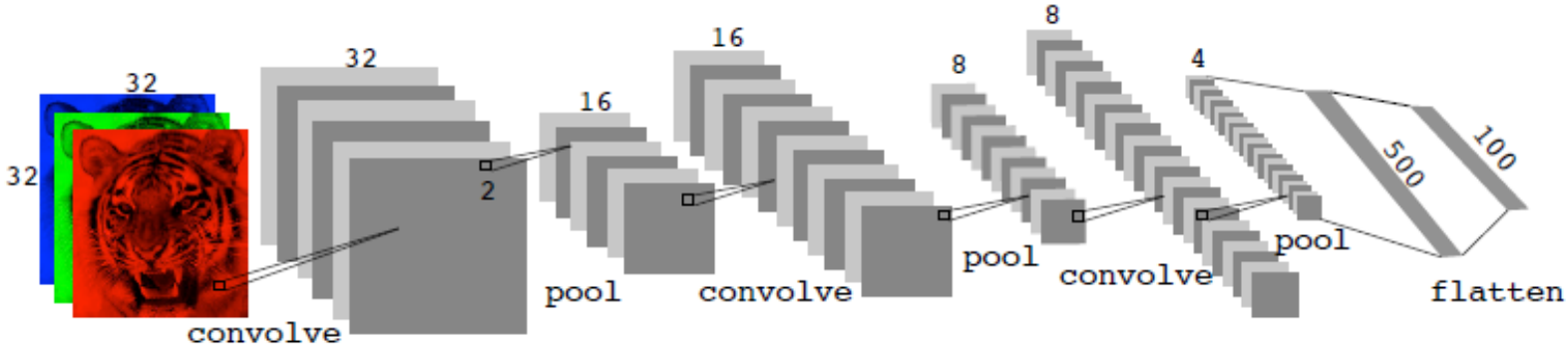

Figura 11: .

Redes neuronales recurrentes RNN

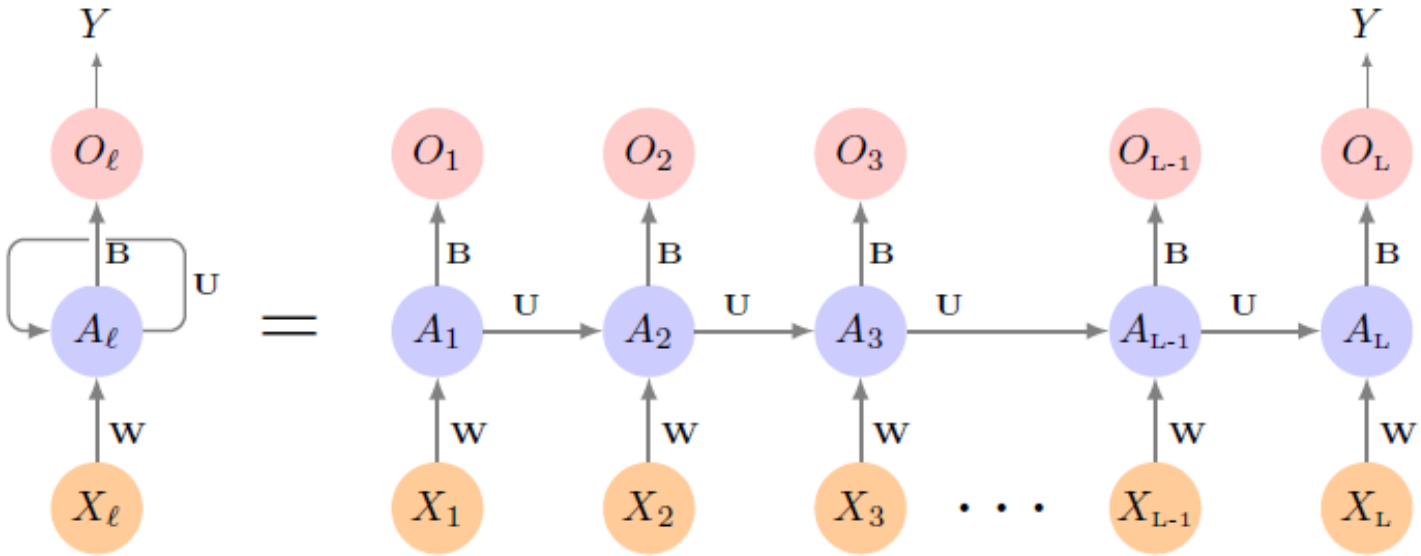

Figura 12: .

- La entrada es una secuencia $\{X_1, X_2, \ldots, X_L\}$. cada X_ℓ es una (palabra) o vector p-dimensional y una sola salida Y.
- La capa oculta también viene representada por una secuencia $\{A_\ell\}_1^L = \{A_1, \ldots, A_L\}$. Acá cada A_ℓ es un vector de dimensión K. $A_\ell^T = (A_{\ell 1}, \ldots, A_{\ell K})$.
- Se representa la colección de $K \times (p+1)$ por la matrix compartida $\boldsymbol{W}$ de pesos w_{kj}, y similarmente por $\boldsymbol{U}$ la matriz $K \times K$ de pesos u_{ks} para los enlaces entre capas ocultas.
- $\boldsymbol{B}$ es un $K+1$-vector de pesos β_k para la capa de salida.

Relación de recurrencia

$$A_{\ell k} = g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} X_{\ell j} + \sum_{s=1}^{K} u_{ks} A_{\ell-1,s}\right), \tag{22}$$

y la respuesta O_ℓ es calculada como

$$O_\ell = \beta_0 + \sum_{k=1}^{K} \beta_k A_{\ell,k}. \tag{23}$$

Con una data consistente de n pares secuencia\respuesta (x_i, y_i), los parámetros son estimados minimizando la suma de cuadrádos

$$\sum_{i=1}^{n}(y_i - o_{iL})^2 = \sum_{i=1}^{n}\left(\left(\beta_0 + \sum_{k=1}^{K}\beta_k g\left(w_{k0} + \sum_{j=1}^{p} w_{kj} x_{iLj} + \sum_{s=1}^{K} u_{ks} A_{i,L-1,s}\right)\right)\right)^2 \tag{24}$$

Pronóstico de series de tiempo

En la figura (13) se ilustran estadísticas hitóricas de negociación sobre el indice Dow Jones de producción industrial. Se ilustran 3 series de tiempo diarias que cubren el periodo Dic 3, 1962 hasta Dic 31, 1986. Las correspondientes series son:

- `Log trading volume` es la fracción de todas las acciones en circulación que se negocian ese día, en relación con una media móvil de los últimos 100 días en una escala logarítmica.
- `Dow Jones return`. Esta es la diferencia entre el logaritmo del Dow Jones en 2 días hábiles consecutivos.
- `Log volatility`. Esto se basa en los valores absolutos de los movimientos del precio diario.

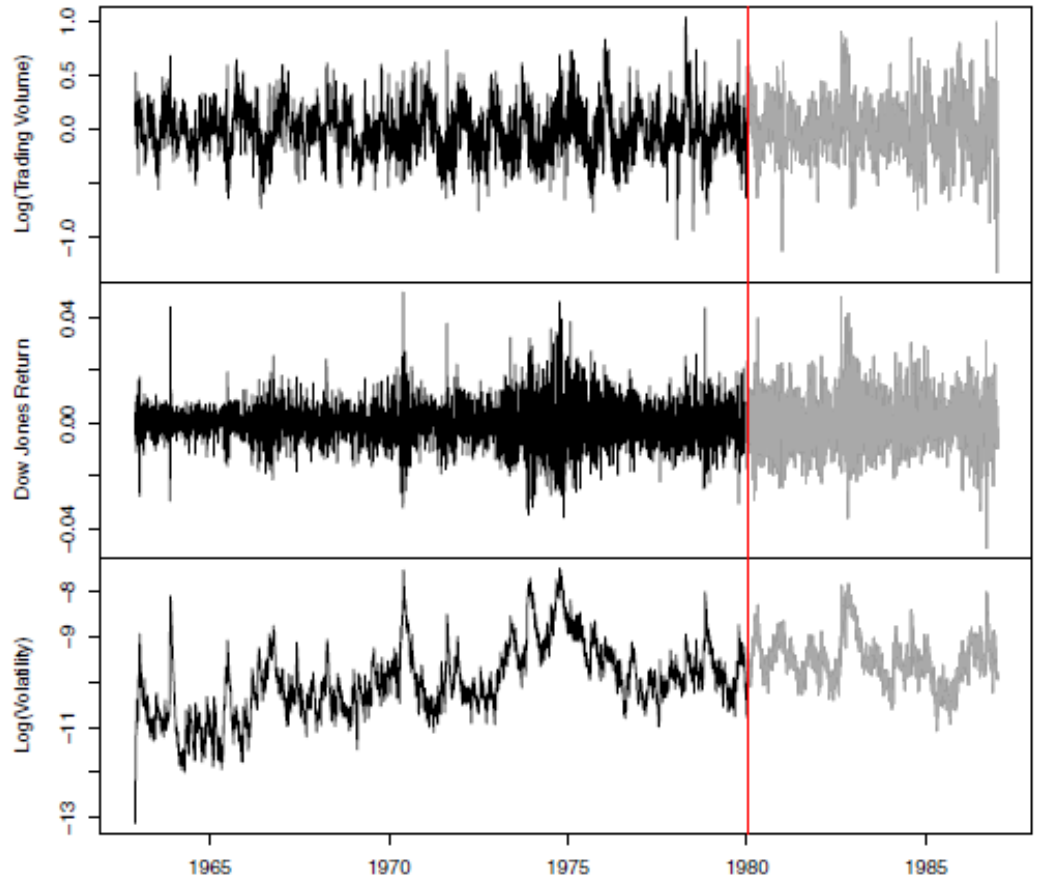

Figura 13: A la derecha de la linea roja (Ene 2,1980) se toman datos de prueba. A la iszquierda de la linea datos de entrenamiento.

La idea consiste en extraer muchas minisecuencias $\{X_1, \ldots, X_L\}$ con un L predefinido que este contexto se denomina 'rezago'. Y una respuesta Y correspondiente

$$X_1 = \begin{pmatrix} v_{t-L} \\ r_{t-L} \\ z_{t-L} \end{pmatrix}, \quad X_2 = \begin{pmatrix} v_{t-L+1} \\ r_{t-L+1} \\ z_{t-L+1} \end{pmatrix}, \cdots, X_L = \begin{pmatrix} v_{t-1} \\ r_{t-1} \\ z_{t-1} \end{pmatrix}, \quad \text{y} \quad Y = v_t. \tag{25}$$

Acá la respuesta objetivo Y es el valor de `log_volume` v_t en un solo instante de tiempo t, y la secuencia de entrada X es la serie de 3-vectores $\{X_\ell\}_1^L$, que consisten en mediciones de `log_volume`,`DJ_return` y `log_volatility`, desde el dia $t - L$, $t - L + 1$, sucesivamente hasta $t - 1$.

- Para cada valor de t desde $L+1$ hasta T se obtiene un par distinto (X, Y).
- Para los datos de `NISE`, se utilizan los últimos 5 días de negociación $(L = 5)$ para predecir el volumen de negociación el próximo día.
- Como $T = 6051$, se obtienen 6046 pares (X, Y).
- En este modelo se emplean K unidades ocultas y se utilizan 4281 datos de entrenamiento para rponósticar 1770 valores de prueba de `log_volume` obteniendose un $R^2 = 0.42$.

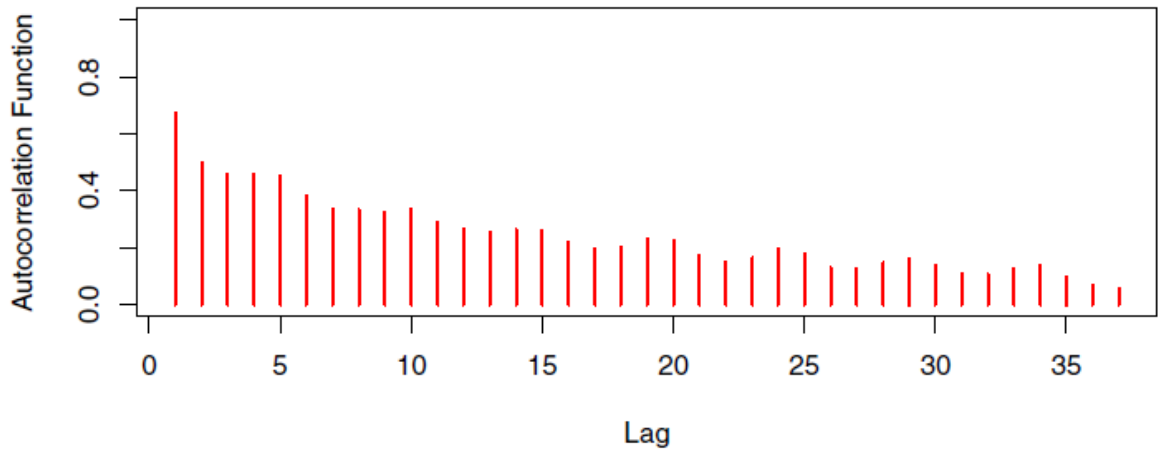

Figura 14: Función de autocorrelación de la variable `log_volume`

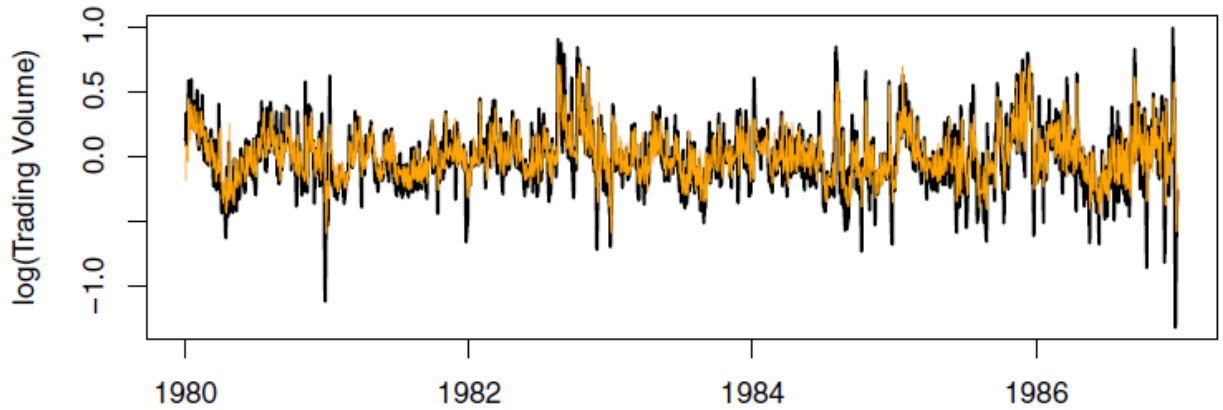

Figura 15

Auto-regresión. AR

La RNN ajustada posee bastante en comun con un tradicional *modelo autoregresivo* (AR). Si se considera únicamente la secuencia de respuesta v_t y se construye un vector respuesta $\boldsymbol{Y}$ y una matriz $\boldsymbol{M}$ de predictores para llevar a cabo una regresión por mínimos cuadrádos, se tiene

$$\boldsymbol{y} = \begin{bmatrix} v_{L+1} \\ v_{L+2} \\ v_{L+3} \\ \vdots \\ v_T \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} 1 & v_L & v_{L-1} & \cdots & v_1 \\ 1 & v_{L+1} & v_L & \cdots & v_2 \\ 1 & v_{L+2} & v_{L+1} & \cdots & v_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{T-1} & v_{T-2} & \cdots & v_{T-L} \end{bmatrix} \tag{26}$$

- $\boldsymbol{M}$ y $\boldsymbol{y}$ cada uno posee $T - L$ filas, una por observación.
- Se aprecia que los predictores para una respuesta v_t en el día t son L valores anteriores de la serie.
- Ajustar una regresión de $\boldsymbol{y}$ en $\boldsymbol{M}$ es equivalente a ajustar el modelo

$$\hat{v}_t = \hat{\beta}_0 + \hat{\beta}_1 v_{t-1} + \hat{\beta}_2 v_{t-2} + \cdots + \hat{\beta}_L v_{t-L}. \tag{27}$$

que se denomina *modelo autoregresivo de orden L* o simplemente AR(L).

Cuando utilizar aprendizaje profundo

- En general, se espera que el aprendizaje profundo sea una opción atractiva cuando el tamaño de la muestra del conjunto de entrenamiento es extremadamente grande, y cuando la interpretabilidad del modelo no es prioridad.
- Cada que sea posible, tiene sentido probar los modelos mś simples y luego hacer una elección basandose en el balance de rendimiento- complejidad.
- En casos en que se tienen varios modelos como conviene seguir el principio de la *cuchilla de Occam* (**Occam Razor**):

  'Ante varios métodos (modelos) que poseen más o menos el mismo desempeño, sellecione el más simple'.