

Análisis de Series de Tiempo

Carrera de Especialización en Inteligencia Artificial

Clase ⁴

Ing. Magdalena Bouza, Ing. Carlos German Carreño Romano

Agenda

- Estimación de parámetros
- Criterios de bondad de modelos
- Diagnóstico de modelos
- Redes Neuronales para series de tiempo
- Caso de estudio

Estimación de parámetros

¿Cómo estimar los parámetros de un modelo?

En general no conocemos los valores de a_1, \dots, a_p y b_1, \dots, b_q y debemos estimarlos a partir de las observaciones de la serie de tiempo.

Existen distintos enfoques:

1. Usando las ecuaciones de Yule-Walker - Sólo sirve para modelos **AR**
2. Cuadrados mínimos (no lo vemos)
3. Método de máxima verosimilitud basado en el modelo de estados

Estimación del modelo AR mediante las ecs. de Y-W

Usando la expresión de las ecs. de Y-W, podemos reemplazar los valores de R_i por \hat{R}_i y resolver el sistema de ecuaciones para a_1, \dots, a_p :

$$\left\{ \begin{array}{l} \hat{R}_1 = a_1 + a_2 \hat{R}_1 + \dots + a_p \hat{R}_{p-1} \\ \vdots \\ \hat{R}_p = a_1 \hat{R}_{p-1} + a_2 \hat{R}_{p-2} + \dots + a_p \hat{R}_p \end{array} \right.$$

Se puede demostrar que los estimadores de Y-W minimizan el ECM.

Una forma eficiente de hacer este cálculo es mediante el algoritmo de Levinson:

1. Set $\hat{\sigma}_0^2 = \hat{C}_0$ and $AIC_0 = N(\log 2\pi \hat{\sigma}_0^2 + 1) + 2$
2. For $m = 1, \dots, M$, repeat the following steps

$$(a) \hat{a}_m^m = \left(\hat{C}_m - \sum_{j=1}^{m-1} \hat{a}_j^{m-1} \hat{C}_{m-j} \right) (\hat{\sigma}_{m-1}^2)^{-1},$$

$$(b) \hat{a}_i^m = \hat{a}_i^{m-1} - \hat{a}_m^m \hat{a}_{m-i}^{m-1} \text{ for } i = 1, \dots, m-1,$$

$$(c) \hat{\sigma}_m^2 = \hat{\sigma}_{m-1}^2 \{1 - (\hat{a}_m^m)^2\},$$

$$(d) AIC_m = N(\log 2\pi \hat{\sigma}_m^2 + 1) + 2(m+1).$$

Estimación del modelo AR por MV

Bajo la suposición de que el ruido $e_i \sim \mathcal{N}(0, \sigma_e^2)$, la serie de tiempos

$$\{Y_1, \dots, Y_n\} \sim \mathcal{N}(0, \Sigma), \quad \Sigma = \begin{bmatrix} C_0 & C_1 & \dots & C_{n-1} \\ C_1 & C_0 & \dots & C_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n-1} & C_{n-2} & \dots & C_0 \end{bmatrix}$$

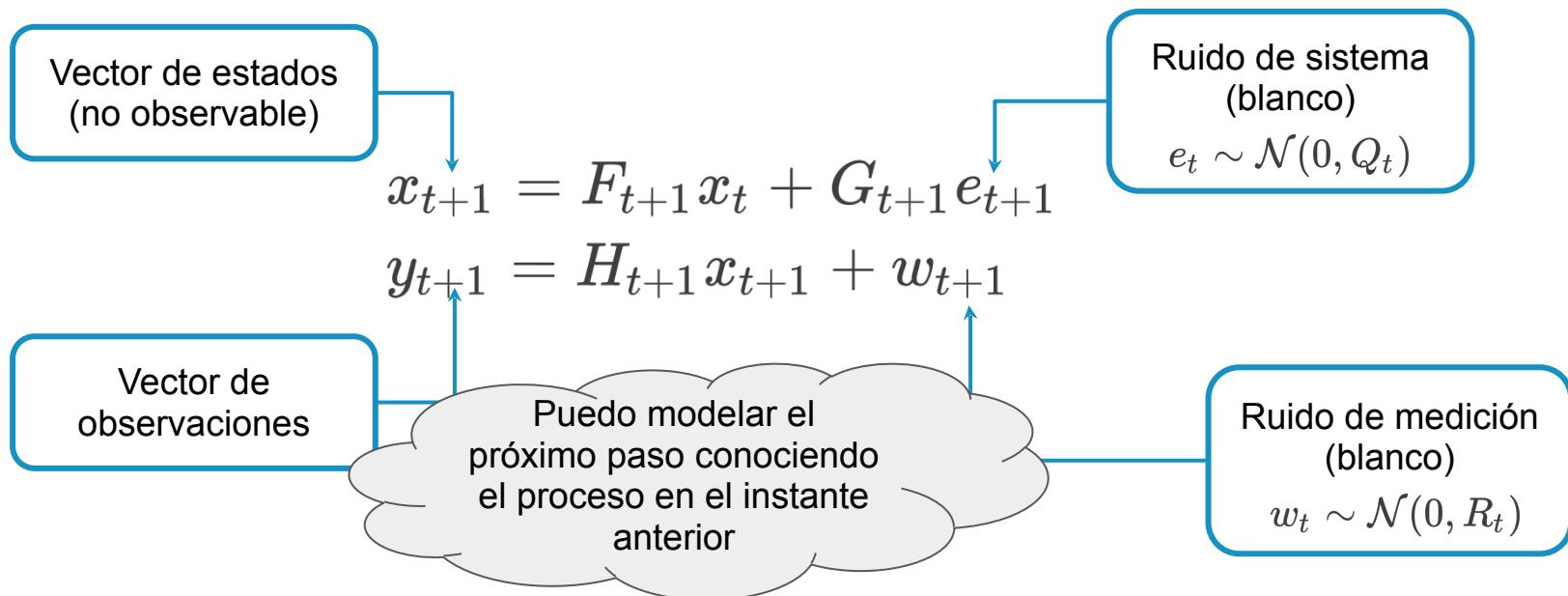
Hallar el EMV puede ser costoso computacionalmente pues hay que calcular Σ^{-1}

Una alternativa es descomponer $L(\theta) = f_{Y_1, \dots, Y_n}(y_1, \dots, y_n; \theta) = \prod_{i=1}^n f(y_i | y_1, \dots, y_{i-1}; \theta)$,
pero ¿cómo calculamos de forma sencilla cada uno de los términos?

$$L(\theta) = f_{\theta}(y_1, \dots, y_n) = f_{\theta}(y_1) f_{\theta}(y_2 | y_1) f_{\theta}(y_3 | y_1, y_2) \dots \\ \theta = (a_1, \dots, a_p, b_1, \dots, b_q)$$

Modelo de estados

Para las series de tiempo, un modelo de estados está dado por



$$f(y_t | y_{t-1}), f(y_{t-1} | y_{t-2}) \dots f(y_2 | y_1)$$

Modelo de estados - Filtro de Kalman

¿Cómo puedo estimar el siguiente paso?

$$Y_t = y_1, y_2, \dots, y_t$$

$$\hat{x}_{t+1|t} = \mathbb{E}[x_{t+1} | Y_t]$$

$$\hat{x}_{t+1|t} = F_t \hat{x}_{t|t}$$

$$P_{t+1|t} = \mathbb{E}[(x_{t+1} - \hat{x}_{t+1|t})(x_{t+1} - \hat{x}_{t+1|t})^T]$$

$$= F_t P_{t|t} A_t^T + B_t Q_t B_t^T$$

$$\hat{x}_{t+1|t+1} = \mathbb{E}[x_{t+1} | Y_{t+1}]$$

$$K_{t+1} = P_{t+1|t} H_{t+1}^T (R_{t+1} + H_{t+1} P_{t+1|t} H_{t+1}^T)^{-1}$$

$$\hat{x}_{t+1|t+1} = \hat{x}_{t+1|t} + K_{t+1} (y_{t+1} - H_{t+1} \hat{x}_{t+1|t})$$

$$P_{t+1|t+1} = \mathbb{E}[(x_{t+1} - \hat{x}_{t+1|t+1})(x_{t+1} - \hat{x}_{t+1|t+1})^T]$$

$$= (I - K_{t+1} H_{t+1}) P_{t+1|t}$$

$$Y_t = y_1, y_2, \dots, y_t$$

Modelo AR(p) como modelo de estados

Si definimos $x_t = (y_t, y_{t-1}, \dots, y_{t-p+1})^T$, nos queda que

$$y_{t+1} = a_1 y_t + a_2 y_{t-1} + \dots + e_{t+1}$$

$$\begin{aligned} x_{t+1} &= F x_t + G e_t \\ \begin{bmatrix} y_{t+1} \\ y_t \\ y_{t-1} \end{bmatrix} &= F \begin{bmatrix} y_t \\ y_{t-1} \\ \vdots \end{bmatrix} + G e_t \end{aligned}$$

$$F = \begin{bmatrix} a_1 & a_2 & \dots & a_p \\ 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 1 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$e_t = (e_t)$$

: Además, considerando $H = [1 \ 0 \ \dots \ 0]$, recuperamos $y_t = Hx_t$. Fijando $Q = \sigma^2$ y $R = 0$ obtenemos el modelo de estados del modelo AR.

Modelo ARMA como modelo de estados

Para el modelo ARMA vamos a definir $x_t = [y_t, y_{t-1}, \dots, y_{t-p+1}]^T$ al igual que en el modelo AR. Las matrices F y H también van a ser las mismas:

$$F = \begin{bmatrix} a_1 & a_2 & \dots & a_p \\ 1 & 0 & \dots & 0 \\ \ddots & & & \\ 1 & 0 & & \end{bmatrix}$$

transición

observaciones

$$H = [1 \ 0 \ \dots \ 0]$$

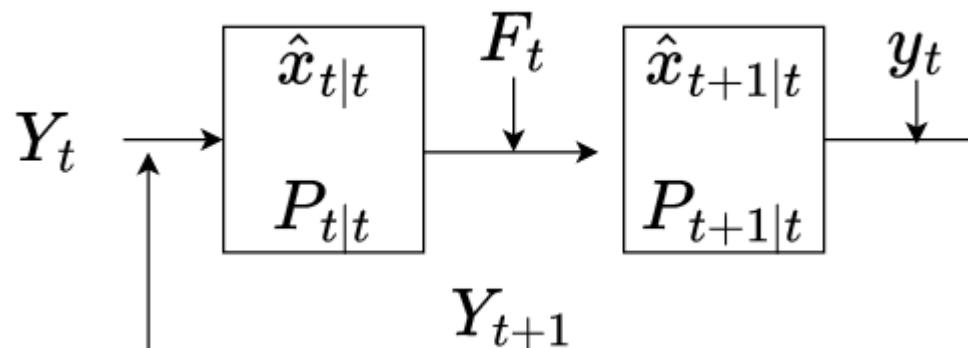
Lo que va a cambiar en el modelo ARMA es la matriz G, que debe incluir la dependencia con ruidos anteriores:

$$G = \begin{bmatrix} 1 & b_1 & \dots & b_q \\ 0 & 0 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & 0 \end{bmatrix}$$
$$e_t = \begin{bmatrix} e_t \\ e_{t-1} \\ \vdots \\ e_{t-q} \end{bmatrix}$$
$$P \times (q+1)$$

Estimación de parámetros con un modelo de estados

Vamos a buscar predecir el vector de estados a partir de las observaciones anteriores, ya que nos va a facilitar para descomponer la verosimilitud.

$$\hat{x}_{t+1|t} = \mathbb{E}[x_{t+1} | \underbrace{y_t, y_{t-1}, \dots, y_1}_{Y_t}] = \mathbb{E}[x_{t+1} | \underline{Y_t}]$$



¿Qué tiene que ver todo esto con MV?

Recordemos que el objetivo es descomponer la verosimilitud como:

$$\begin{aligned}L(\theta) &= f(y_t | y_{t-1}, \dots, y_1; \theta) f(y_{t-1} | y_{t-2}, \dots, y_1; \theta) \dots f(y_2 | y_1; \theta) f(y_1; \theta) \\&= f(y_t | Y_{t-1}; \theta) f(y_{t-1} | Y_{t-2}; \theta) \dots f(y_2 | y_1; \theta) f(y_1; \theta)\end{aligned}$$

O lo que es lo mismo $\log(L(\theta)) = \sum_{i=1}^n \log(f(y_i | Y_{i-1}; \theta))$

El filtro de Kalman me devuelve justamente estas condicionales

Modelos de estados y MV

Además, dado que el ruido $e_t \sim \mathcal{N}(0, Q_t)$, las distribuciones condicionales son de la forma

donde

Serie al tiempo $\rightarrow y_t | Y_{t-1} \sim \mathcal{N}(y_{t|t-1}, d_{t|t-1})$ *↳ predicción de los* *série de* *tiempo*

$$y_{t|t-1} = H_t \hat{x}_{t|t-1} \text{ y } d_{t|t-1} = \underbrace{H_t P_{t|t-1} H_t^T + Q_t}_{\text{Van } \{e\}}$$

Luego

$$\log(L(\theta)) \propto \sum_{i=1}^n d_{t|t-1} + \sum_{i=1}^n (y_t - y_{t|t-1})^T d_{t|t-1}^n (y_t - y_{t|t-1})$$

Cómo usamos el KF + MV para estimar los parámetros

Básicamente los pasos a seguir son:

1. Asignar un valor para θ
2. Dado θ , calcular las matrices del filtro de Kalman
3. Asignar los valores iniciales para $x_{0|0} = 0$, y $P_{0|0}$
4. Para $t=1, \dots, n$ evolucionar el FK y obtener $x_{t|t-1}$, y $P_{t|t-1}$
5. Con estos valores hallar la expresión de MV
6. Actualizar θ y volver a 1.

$$\theta = [a_1, \dots, a_p, b_1, \dots, b_q]$$

Bibliografía y material complementario

- [State-Space modelling](#)
- [Estimating State-Space models through Maximum Likelihood](#)
- “[Kalman Filtering](#)”, Charles K. Chui and Guanrong Chen, Springer
- “[Optimal Filtering](#)”, Brian D.O. Anderson and John B.Moore, Dover

Criterios de Bondad de modelos

Criterios de bondad de modelo

Uno de los objetivos cuando analizamos series de tiempo es poder modelarlas. Esto incluye tanto elegir la familia de modelos correcta, como hallar todos los parámetros de la misma.

Es necesario entonces poder contar con algún criterio que nos permita saber cuán bueno es nuestro modelo, es decir cuán cerca se encuentra la distribución especificada por el modelo a la distribución verdadera de los datos.

Divergencia de Kullback-Leibler (DK-L)

- Mis datos siguen un **proceso desconocido** $g(y)$
- Proponemos un modelo $f(y)$ para aproximarnos a $g(y)$
- "Métrica" para ver si los modelos son parecidos: Divergencia de Kullback-Leibler

$$D_{\text{KL}}(g \parallel f) = \int_{-\infty}^{\infty} g(y) \log\left(\frac{g(y)}{f(y)}\right) dy = \mathbb{E}\left[\log\left(\frac{g(Y)}{f(Y)}\right)\right]$$

Da una medida de la pérdida de información que tenemos al aproximar $g(y)$ por $f(y)$.

Cuanto más pequeño sea el valor de la divergencia K-L, más cerca estarán $f(y)$ y $g(y)$.

El mejor modelo será el que minimice la DK-L.

Estimación de la divergencia K-L

Problema: en general no se conoce el valor verdadero de la distribución $g(y)$,

Solución: Estimamos $D_{\text{KL}}(g \parallel f)$ a partir de las muestras y_1, \dots, y_n .

Supuesto: las muestras son observadas de forma **independiente** de $g(y)$.

En primer lugar, vemos que podemos reescribir

$$D_{\text{KL}}(g \parallel f) = \mathbb{E} \left[\log \left(\frac{g(Y)}{f(Y)} \right) \right] = \boxed{\mathbb{E}[\log(g(Y))]} - \mathbb{E}[\log(f(Y))]$$

 no puede ser calculado (pero
no nos interesa para comparar)

Estimación de la divergencia K-L

Problema: $\mathbb{E}[\log(f(Y))] = \int \log(f(y))g(y)dy$ tampoco se puede calcular.

Solución: la ley de los grandes números garantiza que

$$\underbrace{\frac{1}{n} \sum_{i=1}^n \log(f(y_i))}_{\text{log-verosimilitud}} \rightarrow \mathbb{E}[\log(f(Y))] \\ \ell = \sum_{i=1}^n \log(f(y_i)) = \log(L)$$

Obs: Minimizar $D_{\text{KL}}(g \parallel f)$ equivale a maximizar la log-verosimilitud. Por lo tanto una forma natural de hallar los mejores parámetros para un modelo dado es utilizando el estimador de máxima verosimilitud.

Criterio de Información de Akaike (AIC)

El modelo de información de Akaike sirve para comparar la bondad de dos modelos propuestos, y se basa en los resultados analizados previamente.

$$AIC = 2k - 2\ell(\hat{\theta})$$

cantidad de parámetros parámetros estimados por MV

Se elegirá entonces el modelo que alcance el menor AIC.

Observaciones:

- El resultado de AIC es asintótico pues se basa en la LGN
- AIC no nos dice cuán bueno es cada modelo sino cuál es el mejor de todos (podría ser que sean todos malos)

Criterio de información Bayesiano

El criterio de información Bayesiano también se basa en la log-verosimilitud, pero penaliza más una mayor cantidad de parámetros.

$$\text{BIC} = k \log(n) - 2\ell(\hat{\theta})$$

Nuevamente, se preferirán los modelos con BIC más bajo.

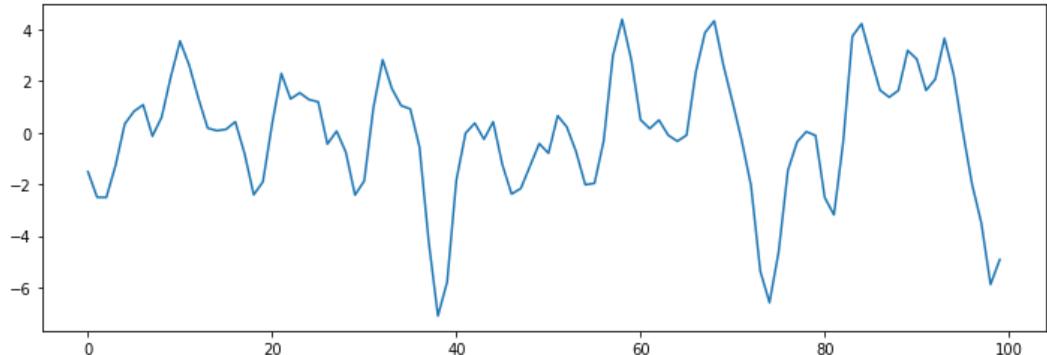
El BIC aparece como consecuencia de un enfoque Bayesiano, donde se busca el modelo f que maximice

$$\mathbb{P}(f|y_1, \dots, y_n)$$

Para una explicación más detallada ver “Elements of Statistical Learning”, Trevor Hastie, Robert Tibshirani, Jerome Friedman.

Ejemplo

```
ar_coef = np.array([0.8,-0.2])
ma_coef = np.array([0.9,0.5,-.3])
arma_ts = arma_generate_sample(
    ar=np.r_[1,-ar_coef], ma=np.r_[1,ma_coef], nsample =100)
plt.figure(figsize=(8,3))
plt.plot(arma_ts)
```



Ejemplo

```
model_bad = ARIMA(arma_ts, order=(5, 0, 5))
model_bad_res = model_bad.fit()
print(model_bad_res.summary())
```

```
model_good = ARIMA(arma_ts, order=(2, 0, 3))
model_good_res = model_good.fit()
print(model_good_res.summary())
```

SARIMAX Results						
Dep. Variable:	y	No. Observations:	100 <th data-cs="3" data-kind="parent"></th> <th data-kind="ghost"></th> <th data-kind="ghost"></th>			
Model:	ARIMA(5, 0, 5)	Log Likelihood	-139.350			
Date:	Sat, 14 May 2022	AIC	302.699			
Time:	12:55:57	BIC	333.961			
Sample:	0 - 100	HQIC	315.351			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.8097	0.432	-1.873	0.061	-1.657	0.037
ar.L1	2.0463	0.526	3.888	0.000	1.015	3.078
ar.L2	-2.0668	0.904	-2.285	0.022	-3.840	-0.294
ar.L3	0.8230	0.966	0.852	0.394	-1.071	2.716
ar.L4	0.1352	0.489	0.277	0.782	-0.822	1.093
ar.L5	-0.1899	0.158	-1.204	0.229	-0.499	0.119
ma.L1	-0.4931	0.554	-0.891	0.373	-1.578	0.592
ma.L2	0.3442	0.172	1.998	0.046	0.006	0.682
ma.L3	-0.0143	0.288	-0.050	0.960	-0.578	0.549
ma.L4	0.6987	0.164	4.250	0.000	0.376	1.021
ma.L5	-0.5023	0.485	-1.035	0.301	-1.453	0.449
sigma2	0.9075	0.139	6.523	0.000	0.635	1.180

SARIMAX Results						
Dep. Variable:	y	No. Observations:	100			
Model:	ARIMA(2, 0, 3)	Log Likelihood	-142.279			
Date:	Sat, 14 May 2022	AIC	298.559			
Time:	12:55:48	BIC	316.795			
Sample:	0 - 100	HQIC	305.939			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
const	-0.8281	0.434	-1.908	0.056	-1.679	0.023
ar.L1	1.1745	0.620	1.894	0.058	-0.041	2.390
ar.L2	-0.3519	0.225	-1.565	0.118	-0.793	0.089
ma.L1	0.4292	0.647	0.663	0.507	-0.839	1.697
ma.L2	-0.0531	0.792	-0.067	0.946	-1.605	1.499
ma.L3	-0.6309	0.574	-1.100	0.271	-1.755	0.494
sigma2	0.9665	0.139	6.963	0.000	0.694	1.239
Ljung-Box (L1) (Q):			0.02	Jarque-Bera (JB):		0.32
Prob(Q):			0.88	Prob(JB):		0.85
Heteroskedasticity (H):			1.09	Skew:		-0.14
Prob(H) (two-sided):			0.80	Kurtosis:		2.96

Diagnóstico de modelos

Diagnóstico de los modelos

Se basa en determinar la bondad de la estimación del modelo, y en caso de tener un mal ajuste proponer modificaciones apropiadas.

Vamos a ver dos enfoques complementarios:

- Análisis residual
- Análisis de modelos sobre-parametrizados

1. Análisis de residuos

Al igual que en los problemas de regresión, vamos a llamar residuo a la diferencia entre el valor verdadero y el estimado por el modelo:

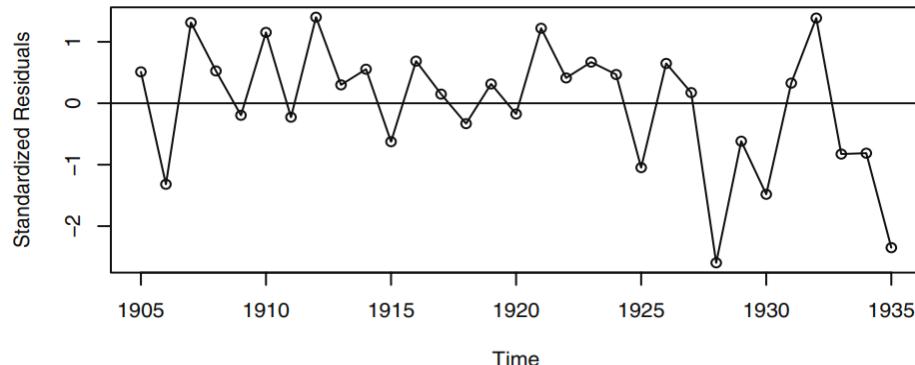
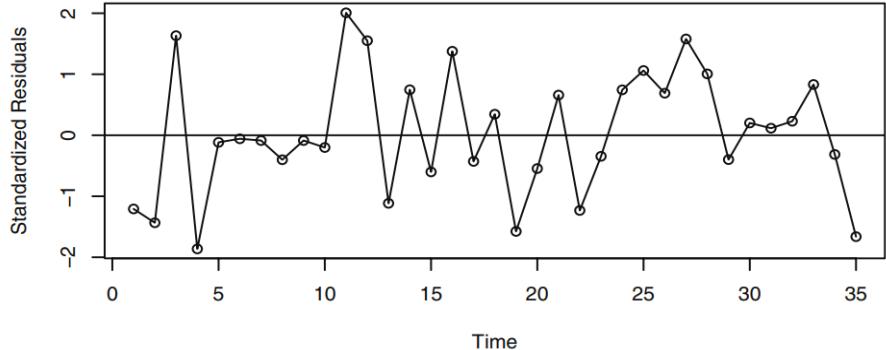
$$\hat{e}_t = Y_t - \hat{Y}_t$$

Si el modelo se encuentra bien estimado, los residuos deberían tener aproximadamente las propiedades del ruido blanco.

Las desviaciones de esta propiedad pueden ayudarnos a corregir el modelo.

Gráfico de residuos

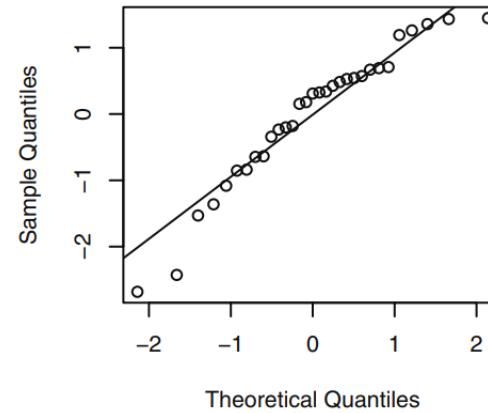
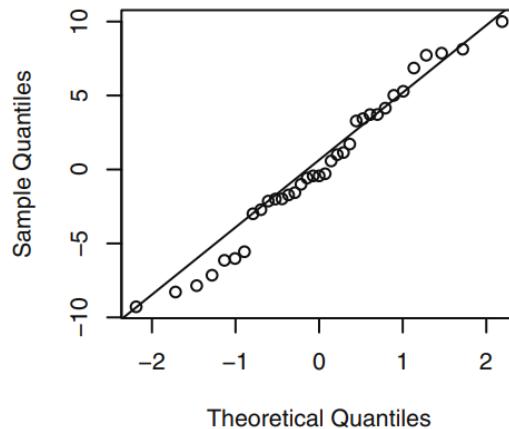
El primer testeo de diagnóstico consiste en graficar los residuos obtenidos a lo largo del tiempo.



Normalidad de los residuos

Una buena forma de analizar la normalidad de los residuos es mediante un QQplot.

También pueden analizarse tests estadísticos como el Shapiro-Wilk (recordar que si la cantidad de muestras es muy grande este camino puede no ser recomendable).

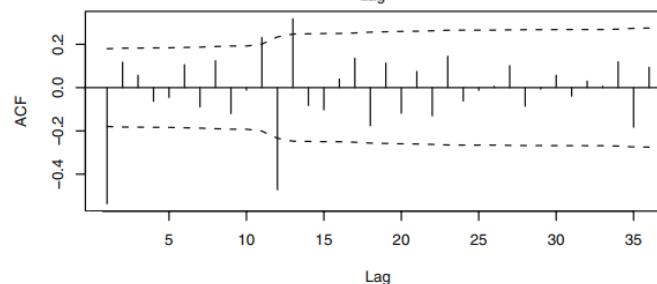
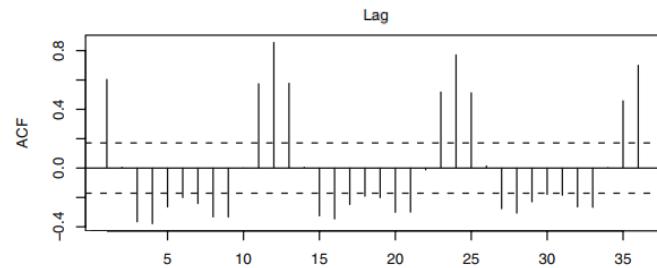
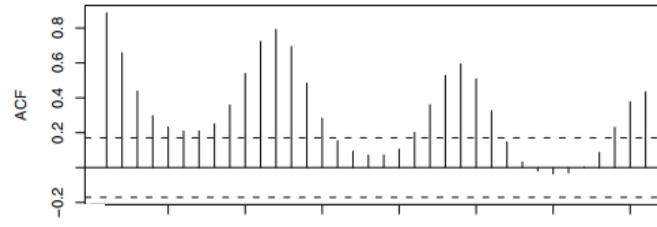
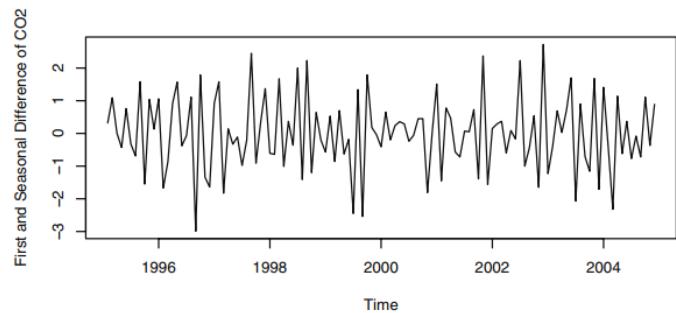
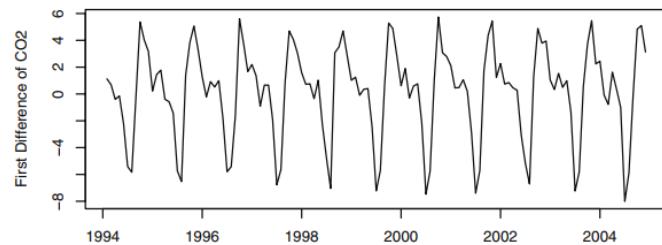
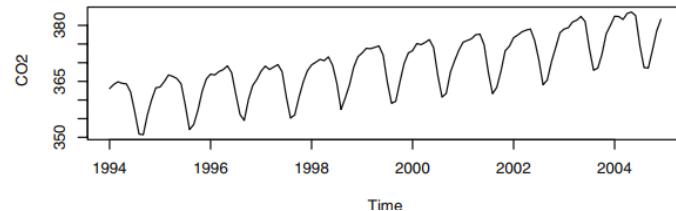


Autocorrelación de los residuos

Para analizar la independencia de los residuos podemos estimar su función de autocorrelación.

Idealmente, para tamaños grandes de muestra, las estimaciones de las correlaciones seguir una distribución normal de media 0 y varianza $1/n$. Sin embargo esto no resulta del todo cierto para lags pequeños j, k , donde la varianza puede resultar significativamente menor y estar altamente correlacionados

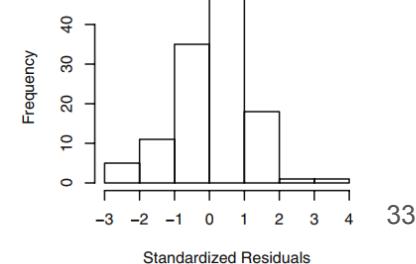
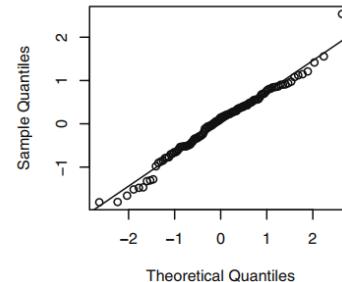
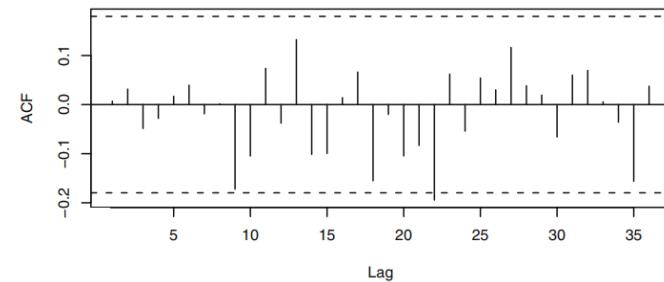
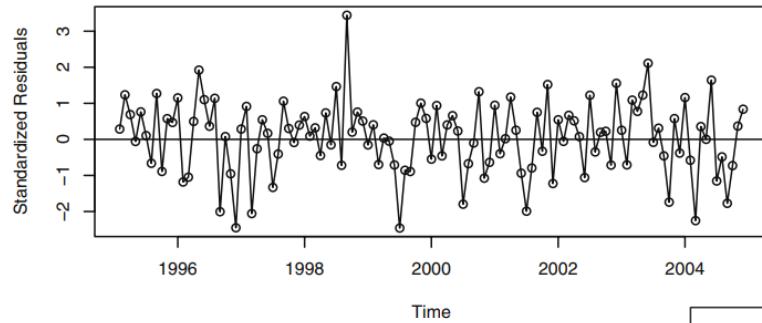
Especificación del modelo - ejemplo



Ajuste del modelo

Una vez especificado el modelo (en el ejemplo anterior un candidato sería SARIMA(0,1,1)x(0,1,1)₁₂) debemos ajustar los parámetros.

Nuevamente vamos a analizar los residuos de la estimación



Test de Ljung-Box

Contempla las magnitudes de estas **autocorrelaciones en conjunto**.

Box y Pierce propusieron el estadístico $Q = n(\hat{r}_1^2 + \hat{r}_2^2 + \dots + \hat{r}_k^2)$. Mostraron que si los órdenes p y q del ARMA están bien estimados, y n es grande, $Q \approx \chi_{k-p-q}^2$. El problema es que la dist. asintótica está basada en un teorema límite.

Ljung y Box demostraron que esta dist. no se cumple para tamaños comunes de muestras. Propusieron $Q_* = n(n+2)(\frac{\hat{r}_1^2}{n-1} + \frac{\hat{r}_2^2}{n-1} + \dots + \frac{\hat{r}_k^2}{n-1})$ que se asemeja mucho más a la distribución chi2.

En ambos casos el test asociado es **H0: “Los residuos están descorrelacionados”**. Busco **no rechazar** la hipótesis nula

2. Overfitting y redundancia de parámetros

Cuando tenemos un modelo ARMA, se presenta el problema de redundancia de parámetros o falta de identificabilidad. Si $a(B)Y_t = b(B)e_t$ es el modelo correcto, luego también es correcto el modelo $(1 - cB)a(B)Y_t = (1 - cB)b(B)e_t$ para cualquier constante c . Sin embargo, si el modelo original se correspondía con un ARMA(p,q), el segundo es un ARMA(p+1,q+1). Decimos en este caso que hay redundancia de parámetros.

1. Especificamos el modelo más sencillo que se vea factible (antes de probar alguno más complejo)
2. Al hacer overffiting, agrandamos la parte MA y la AR por separado
3. Expandir el modelo en la dirección sugerida por el análisis de residuos.

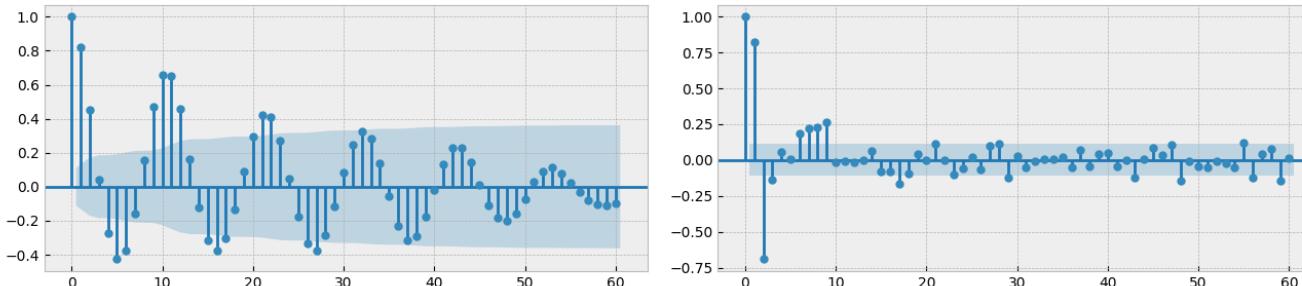
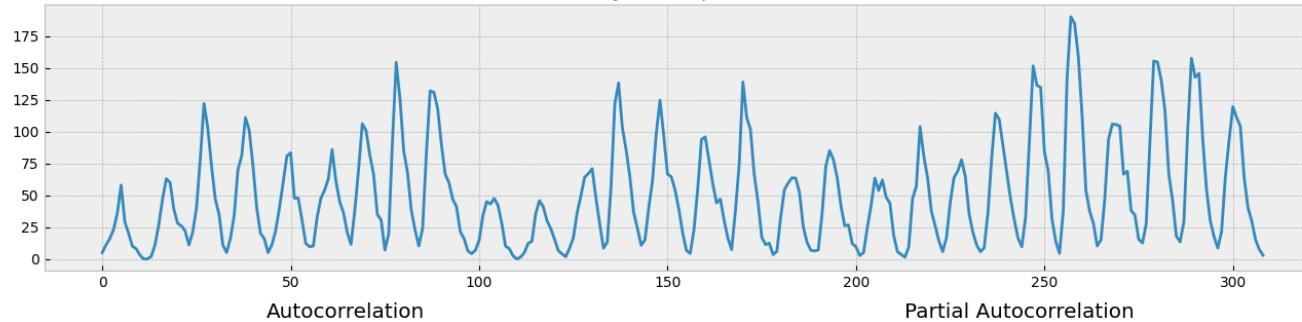
Resultados (S) ARIMA

```
1                               SARIMAX Results
2 =====
3 Dep. Variable:                      y     No. Observations:                 4840
4 Model:                 ARIMA(5, 1, 0)   Log Likelihood:            -10901.009
5 Date:                Thu, 11 Nov 2021   AIC:                         21814.019
6 Time:                     17:51:37      BIC:                         21852.926
7 Sample:                   0 - 4840    HQIC:                        21827.678
8 Covariance Type:            opg
9
10 =====
11              coef    std err        z      P>|z|    [ 0.025    0.975]
12 -----
13 ar.L1      0.0367    0.004    9.613    0.000     0.029    0.044
14 ar.L2      0.0218    0.005    4.612    0.000     0.013    0.031
15 ar.L3      0.0451    0.005    9.124    0.000     0.035    0.055
16 ar.L4     -0.0205    0.005   -3.941    0.000    -0.031   -0.010
17 ar.L5      0.0056    0.005    1.041    0.298    -0.005    0.016
18 sigma2     5.2995    0.027  199.035    0.000     5.247    5.352
19
20 Ljung-Box (L1) (Q):             0.00  Jarque-Bera (JB):          227927.17
21 Prob(Q):                      1.00  Prob(JB):                  0.00
22 Heteroskedasticity (H):       953.07  Skew:                      -0.30
23 Prob(H) (two-sided):           0.00  Kurtosis:                  36.62
24
```

Volvamos al ejemplo de sun activity

Serie de actividad del sol (SUNACTIVITY)

Time Series Analysis Plots
Dickey-Fuller: $p=0.05308$

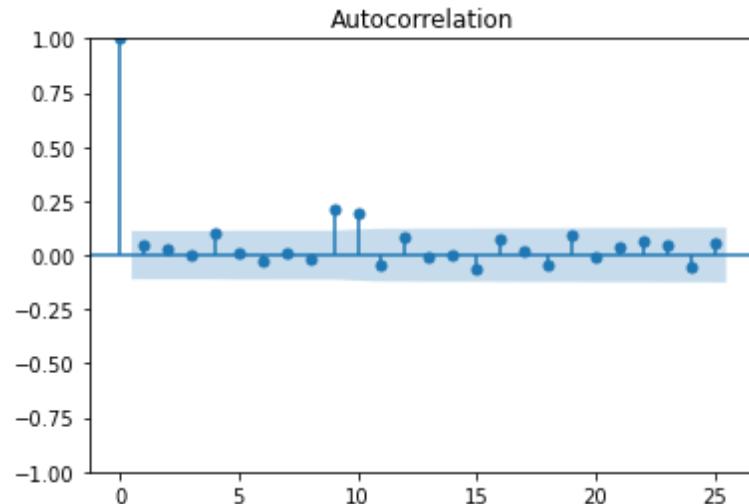
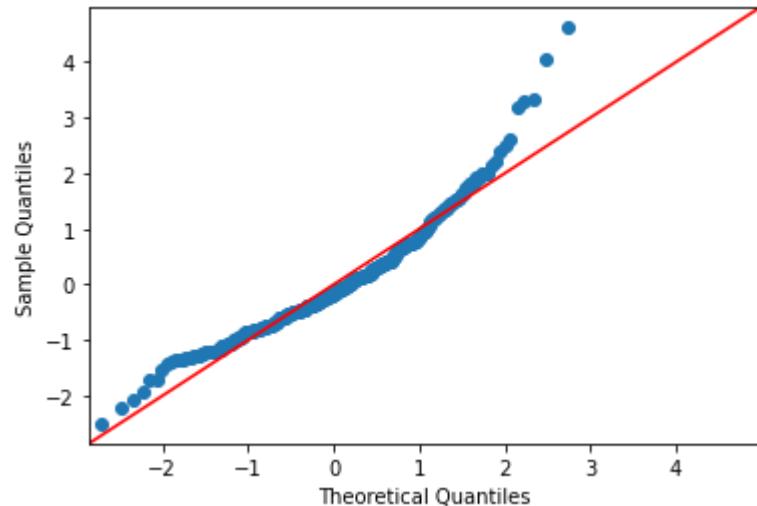


Ajuste de modelo SARIMAX(2,0,3)x(1,1,1)10

No. Observations:	309
Log Likelihood	-1257.865
AIC	2531.730
BIC	2561.307
HQIC	2543.569

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	1.2853	0.183	7.037	0.000	0.927	1.643
ar.L2	-0.6071	0.108	-5.632	0.000	-0.818	-0.396
ma.L1	-0.0998	0.187	-0.533	0.594	-0.467	0.267
ma.L2	0.0945	0.138	0.685	0.493	-0.176	0.365
ma.L3	0.0224	0.122	0.184	0.854	-0.217	0.261
ar.S.L11	0.2089	0.074	2.826	0.005	0.064	0.354
ma.S.L11	-0.9780	0.144	-6.787	0.000	-1.260	-0.696
sigma2	247.3620	30.231	8.182	0.000	188.110	306.614
Ljung-Box (L1) (Q): 0.08				Jarque-Bera (JB): 155.97		
Prob(Q): 0.78				Prob(JB): 0.00		
Heteroskedasticity (H): 1.43				Skew: 1.12		
Prob(H) (two-sided): 0.08				Kurtosis: 5.74		

¿Qué pasa con los residuos?



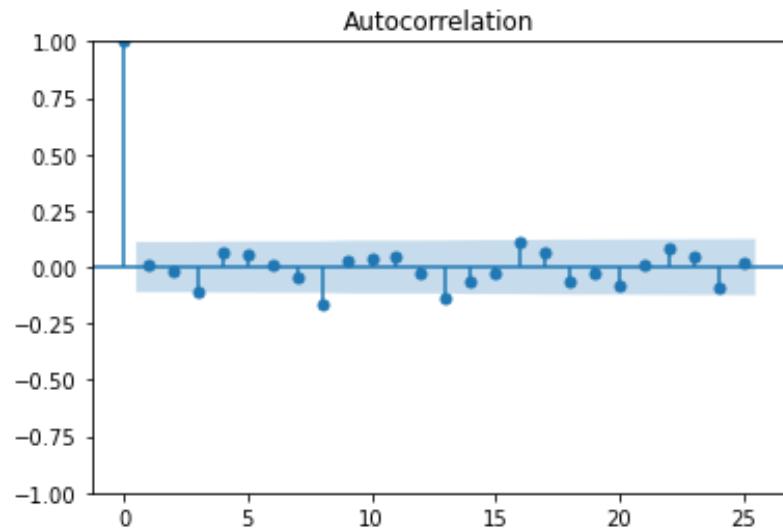
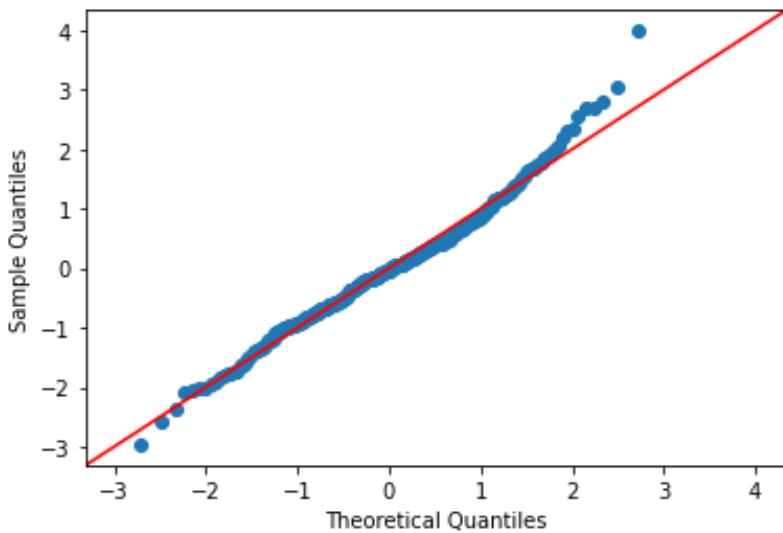
El modelo
todavía se
puede mejorar

Probemos con otro: SARIMA(2,0,3)x(0,1,1)10

No. Observations:	309
Log Likelihood	-1251.934
AIC	2517.868
BIC	2543.701
HQIC	2528.211

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	1.6084	0.029	55.070	0.000	1.551	1.666
ar.L2	-0.9328	0.024	-38.205	0.000	-0.981	-0.885
ma.L1	-1.4432	0.061	-23.733	0.000	-1.562	-1.324
ma.L2	0.4691	0.101	4.651	0.000	0.271	0.667
ma.L3	0.1220	0.064	1.902	0.057	-0.004	0.248
ma.S.L12	-0.9987	2.759	-0.362	0.717	-6.406	4.409
sigma2	241.3748	662.225	0.364	0.715	-1056.563	1539.313
Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	24.02			
Prob(Q):	0.97	Prob(JB):	0.00			
Heteroskedasticity (H):	1.41	Skew:	0.42			
Prob(H) (two-sided):	0.09	Kurtosis:	4.11			

Probemos con otro: SARIMA(2,0,3)x(1,1,1)10



¡Este se ve
mejor!

Redes Neuronales para Series de Tiempo

Redes Neuronales

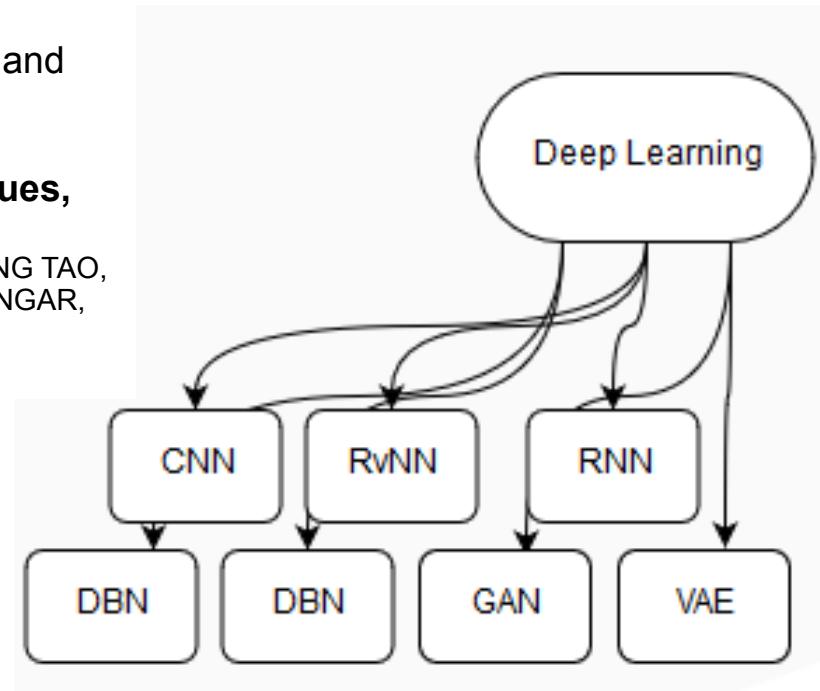
Bibliografía

- **Deep Learning**, Ian Goodfellow and Yoshua Bengio and Aaron Courville
- **A Survey on Deep Learning: Algorithms, Techniques, and Applications**

SAMIRA PUYANFAR, SAAD SADIQ, YILIN YAN, HAIMAN TIAN, YUDONG TAO, MARIA PRESA REYES, MEI-LING SHYU, SHU-CHING CHEN, S.S.IYENGAR, ACM Computing Surveys, Vol. 51, No. 5, Article 92. **September 2018**

Frameworks

- Caffe, DL4j, Torch, Neon, Theano, MXNet, TensorFlow, CNTK



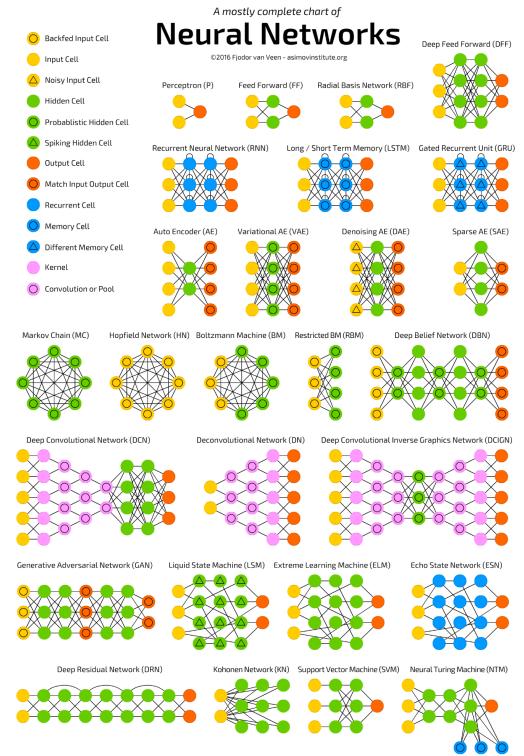
Redes Neuronales Recurrentes y Recursivas

10

Sequence Modeling: Recurrent and Recursive Nets

Recurrent neural networks, or RNNs (Rumelhart et al., 1986a), are a family of neural networks for processing sequential data. Much as a convolutional network is a neural network that is specialized for processing a grid of values \mathbf{X} such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)}$. Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length.

Ref.: Yoshua Bengio, Deep Learning Adaptive Computation and Machine Learning



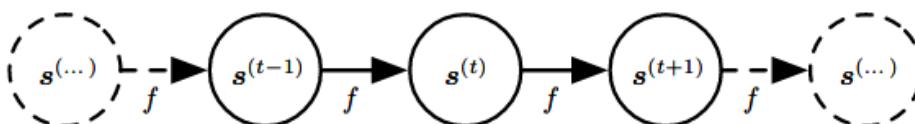
Redes Neuronales Recurrentes

$$y_t = \mu_t + x_t$$

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

- $s^t(t)$ es el **estado** del sistema
- la definición recurrente se refiere a $t-1$
- la función $f(s, \theta)$ mapea estados

$$\begin{aligned} s^{(3)} &= f(s^{(2)}; \theta) \\ &= f(f(s^{(1)}; \theta); \theta) \end{aligned}$$



$$s^{(t)} = f(s^{(t-1)}, x^{(t)}, \theta)$$

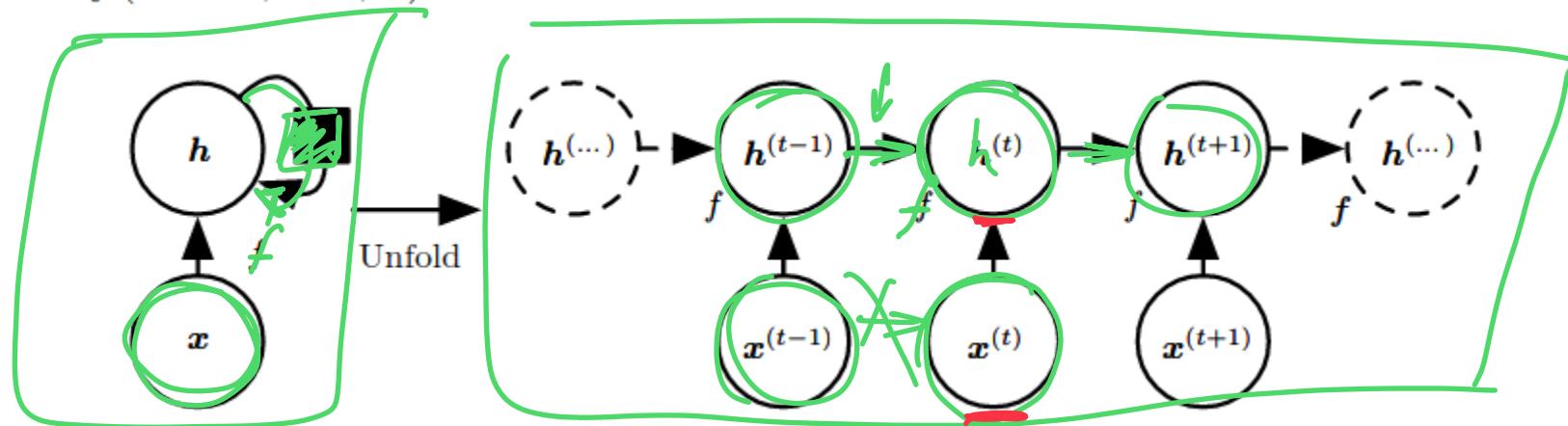
- $x^t(t)$ es una señal externa
- la definición recurrente se refiere a $t-1$

Redes Neuronales Recurrentes

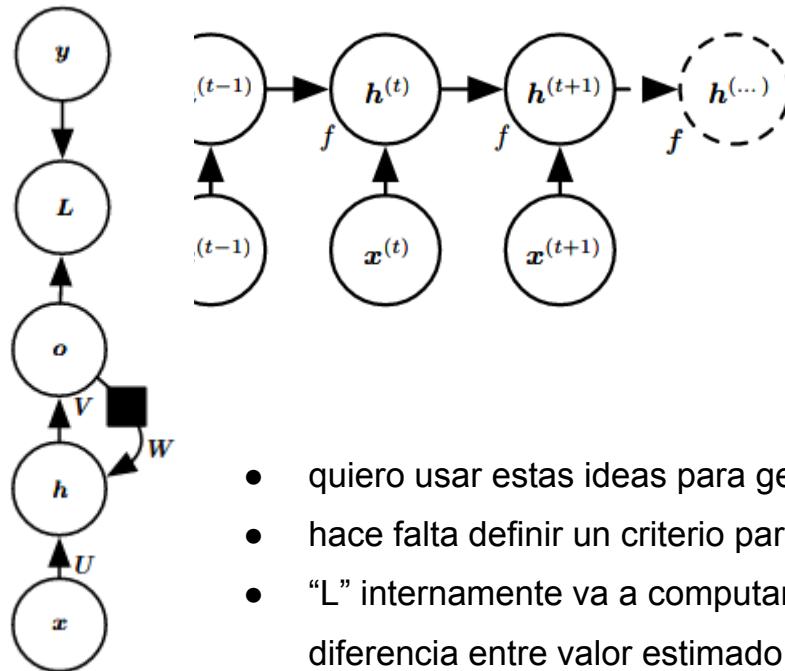
$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta)$$

$$\begin{aligned}\mathbf{h}^{(t)} &= g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)}, \dots, \mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \\ &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta).\end{aligned}$$

- Usamos $\mathbf{h}^{(t)}$ para representar un **estado oculto** (hidden) del sistema
- Una notación gráfica compacta simplifica la vista desplegada de recurrencia
- $g^{(t)}$ usa toda la serie para el estado $s(t)$



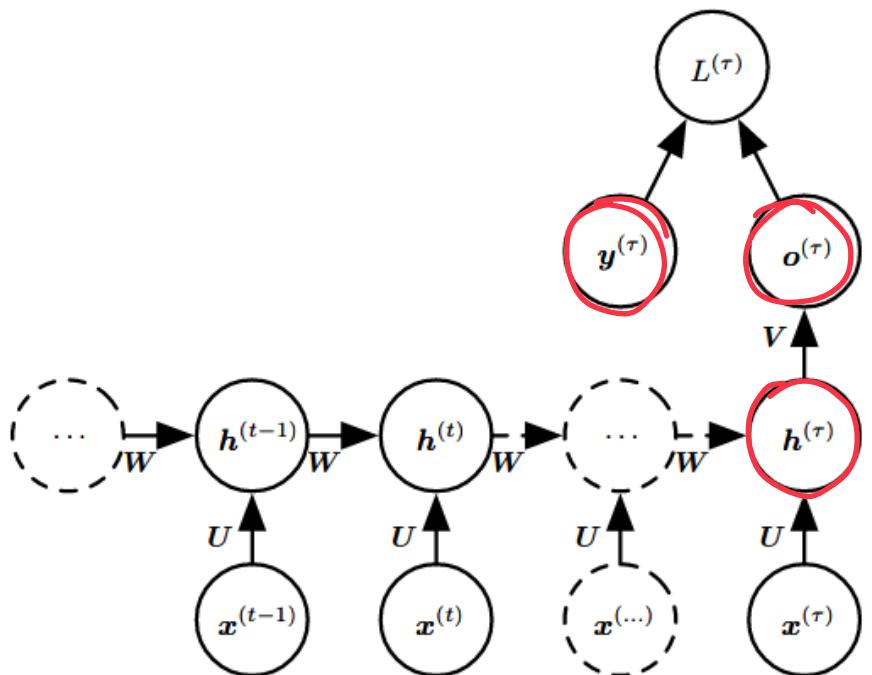
Redes Neuronales Recurrentes



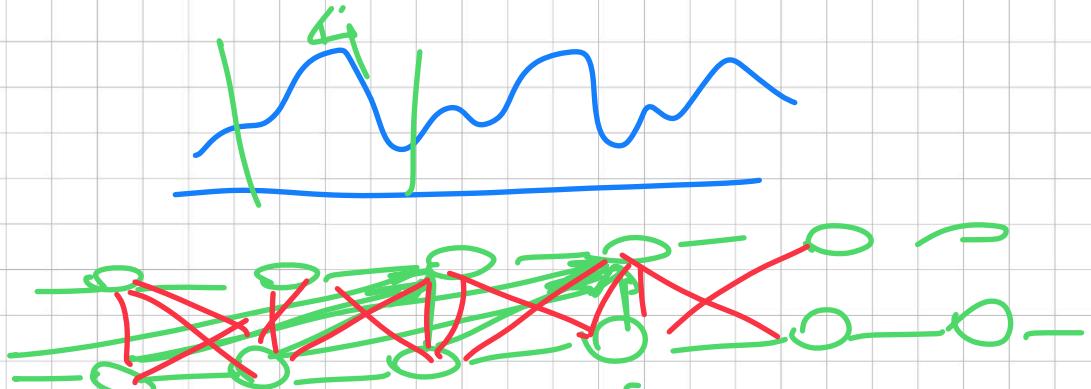
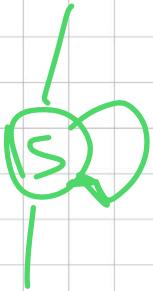
- hay una ventaja en esta notación que refiere al largo de la red: no es el largo de la serie, sino de transiciones de estados.
- es posible usar la misma función de transición f con los mismos parámetros en cada paso.
Se posibilita **generalización**.

- quiero usar estas ideas para generar una salida $o^{\wedge}(t)$
- hace falta definir un criterio para optimizar esa salida usando una función de costo L (loss)
- “ L ” internamente va a computar un estimador $\hat{y}^{\wedge}(t)$ para $y^{\wedge}(t)$ y va a buscar minimizar la diferencia entre valor estimado y valor obtenido.

Redes Neuronales Recurrentes

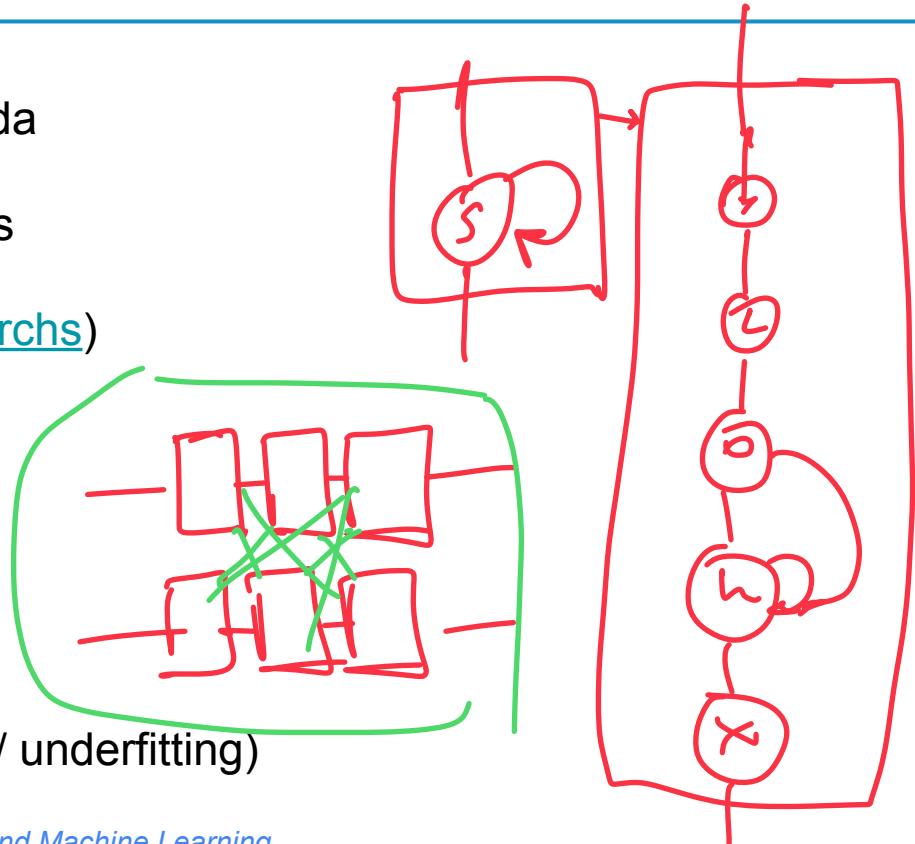


- Distintas **topologías** producen distintos resultados
- RNN que produce una salida en cada paso a partir de conexiones entre unidades ocultas
- RNN que produce una salida en cada paso a partir de sólo el estado anterior
- RNN con conexiones ocultas que leen toda la secuencia y producen una sola salida
- RNN con múltiples salidas... (próx. clase)



Terminología

- Celdas de entrada, ocultas y de salida
- Perceptrones o unidades funcionales
- Circuitos, topologías, arquitectura ([archs](#))
- Función de costo
- Funciones de activación
- Entrenamiento / Test
- Sobreajuste / sub-ajuste (overfitting / underfitting)



Ejemplo con Torch

- Identificar las etapas del algoritmo
 - dataset
 - formato
 - definir arquitectura (input, hidden, output)
 - inicializar
 - entrenar, iterar (epochs)
 - función de costo
 - predicción como etapa de test

Caso de estudio
Pronósticos de la demanda de
capacidad de **Internet**

Sizing Techniques applied to Network Capacity Planning

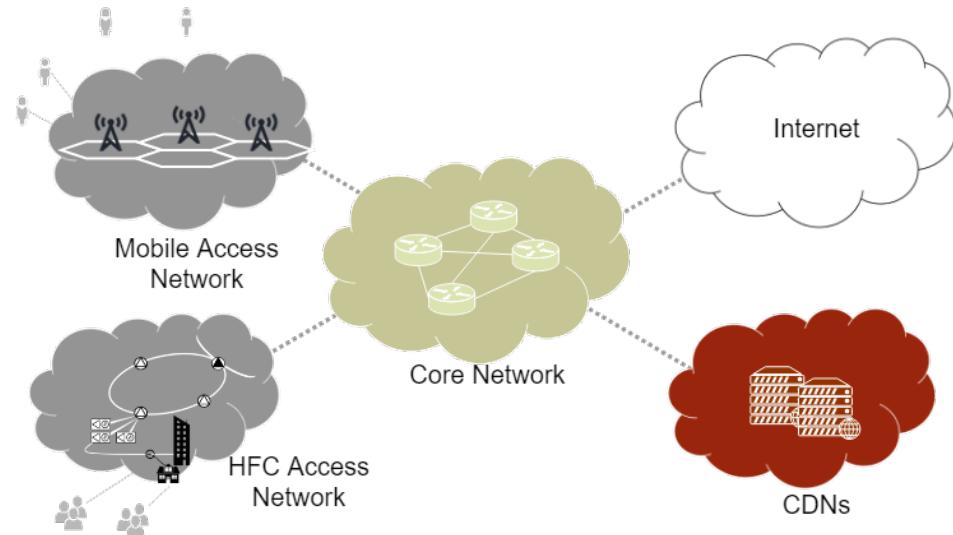
Técnicas de dimensionamiento aplicadas al planeamiento de capacidad de red

- Carlos G. Carreño Romano
- Natalia A. Clivio Velilla

<https://ieeexplore.ieee.org/document/8646077>

Aplicaciones en dimensionamiento de redes

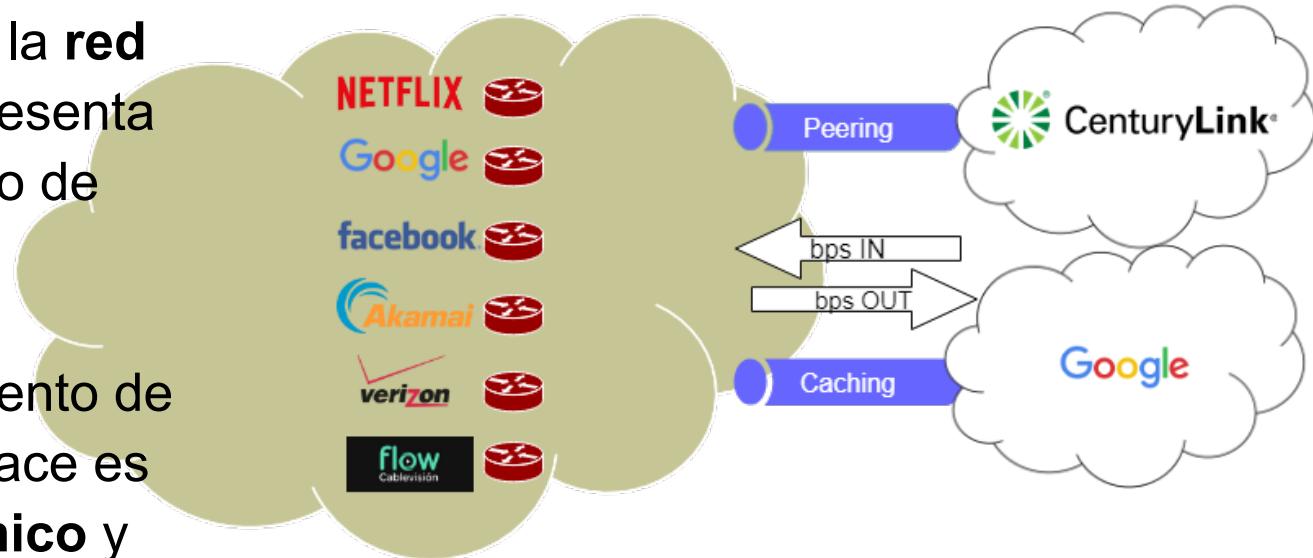
- Proveedores de Servicios de Comunicaciones (CSP)
- Redes de acceso
- Redes Core
- Redes de Distribución de Contenidos (CDN)
- Internet



Contexto

Cada enlace entre la **red Core** y las **CDN** presenta una serie de tiempo de tráfico.

Predecir el crecimiento de tráfico en cada enlace es de interés **económico** y **técnico**.



Dataset

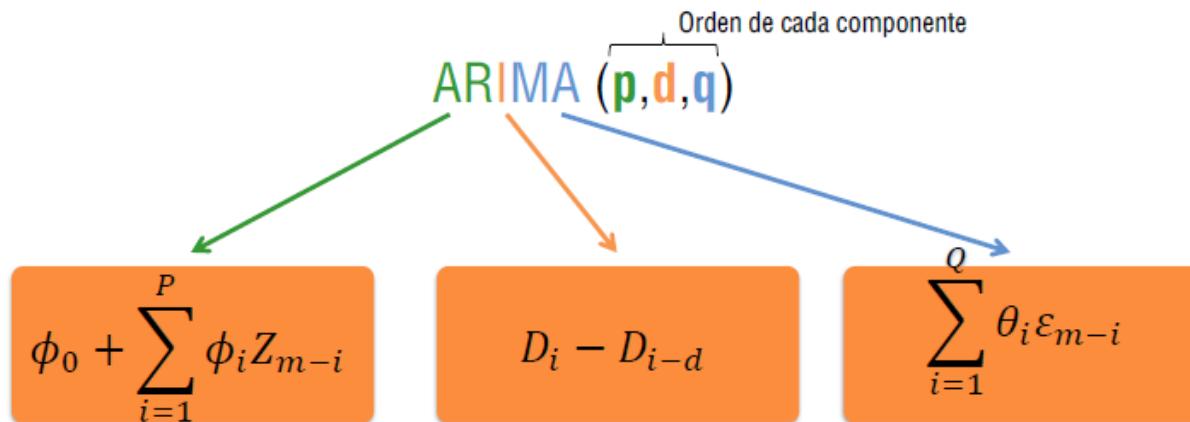


Datos de 1 año con una granularidad de 5 minutos.

Objetivo: Comparar predicciones generadas con ARIMA y LSTM RNN para las trazas de tráfico real de la empresa Telecom Argentina

Modelado usando ARIMA

ARIMA viene de Autorregresión Integrados de Media móvil



Autoregresivo de los
últimos 'p' valores

Diferenciación de 'd'
períodos anteriores

Promedio móvil de los
últimos 'q' errores

Modelado usando ARIMA

Para aplicar modelos ARIMA se suele descomponer la serie, analizando en primer lugar la **tendencia** de la serie, luego la **estacionalidad** y concentrándose en identificar estas componentes filtradas. Hay otras dos componentes que hacen el modelo completo y son las **componentes cíclicas** y las **componentes aleatorias**. El proceso consiste en la descomposición de la serie en forma aditiva o multiplicativa. Usamos en este trabajo la forma aditiva y definimos entonces una serie de tiempo $Y(t)$ como:

$$Y(t) = T(t) + S(t) + C(t) + e(t)$$

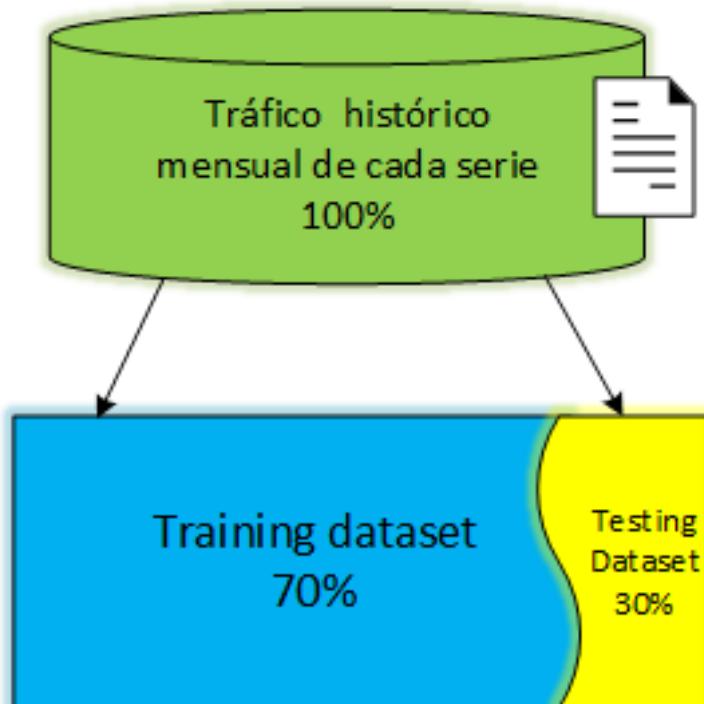
donde:

para tiempo continuo y una serie de tiempo discreto Y_t como:

$$Y_t = T_t + S_t + C_t + e_t$$

- $T(t)$: Tendencia
- $S(t)$: Variación Estacional
- $C(t)$: Componente Cíclica
- $e(t)$: Componente aleatoria

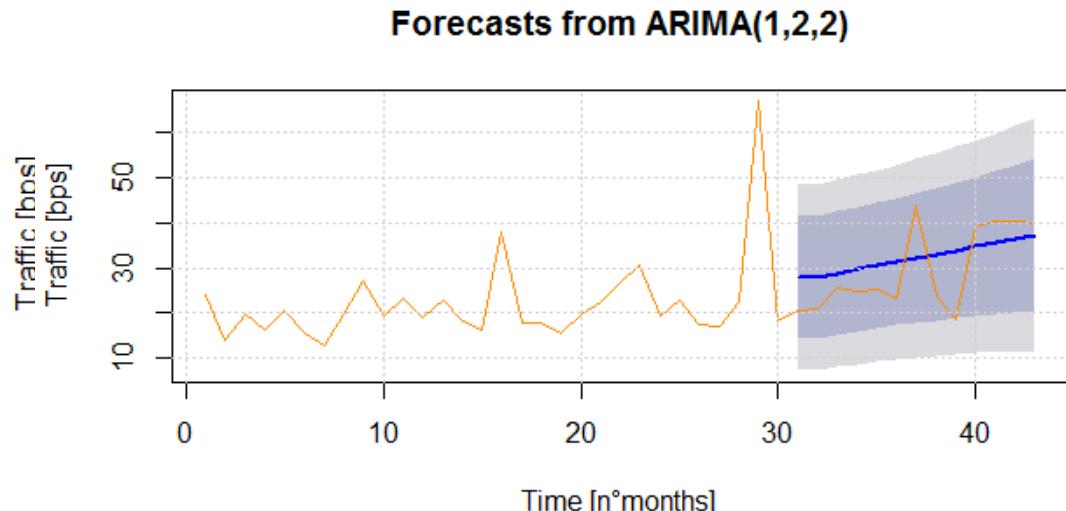
Training and Testing



- Hace falta fraccionar el dataset en dos partes: **training y test**
- En general es útil que las proporciones sean representativas.
- La cantidad y calidad de los datos es un factor siempre presente.
- No todos los algoritmos estadísticos admiten paralelismo.
- Otro factor importante es el nivel de ajuste (**sub fitting vs. overfitting**)

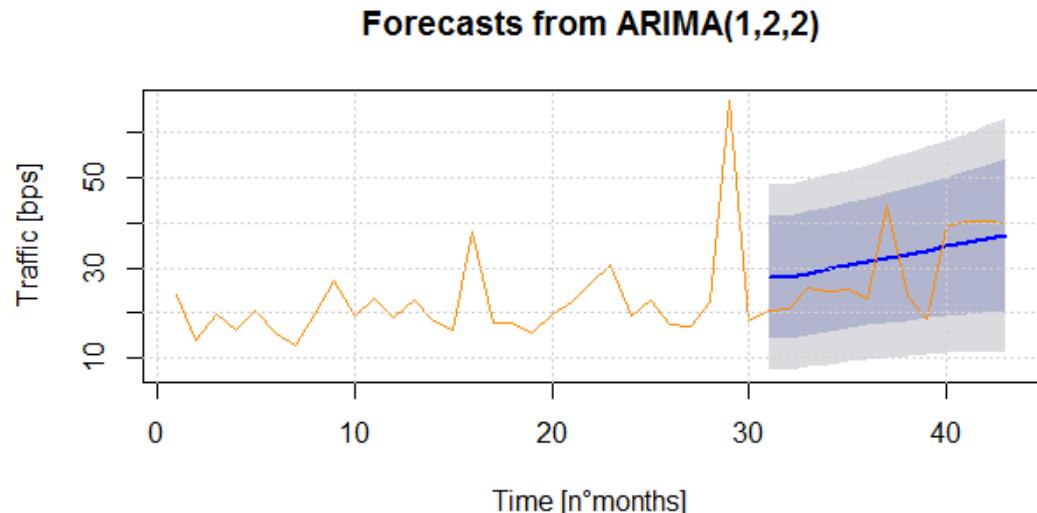
Resultados usando ARIMA

- En azul se grafica la **tendencia**
- En gris oscuro el intervalo de confianza del 95%
- En gris claro el intervalo e 90%
- En naranja el fragmento de testing de la serie



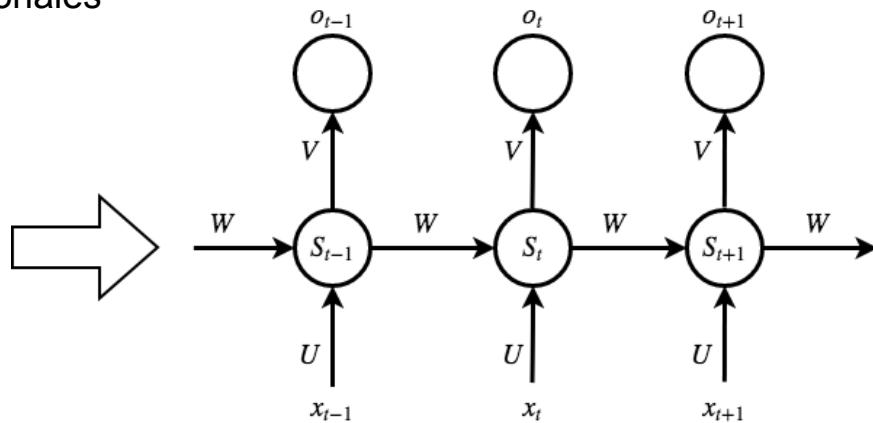
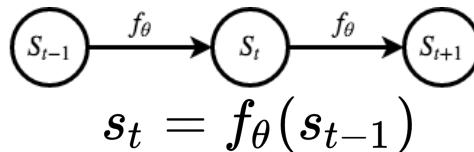
Compromisos

- Los intervalos de confianza pueden ser muy amplios en términos absolutos.
- Si es una variable económica el desvío puede ser demasiado significativo.



Alternativa usando Redes Neuronales

- Usamos redes basadas en estados
- este tipo de redes se llaman Redes Neuronales Recurrentes (RNN)

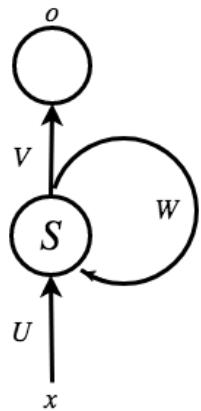
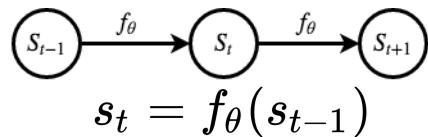


- se puede usar la notación de estados
- se suele usar la topología desplegada

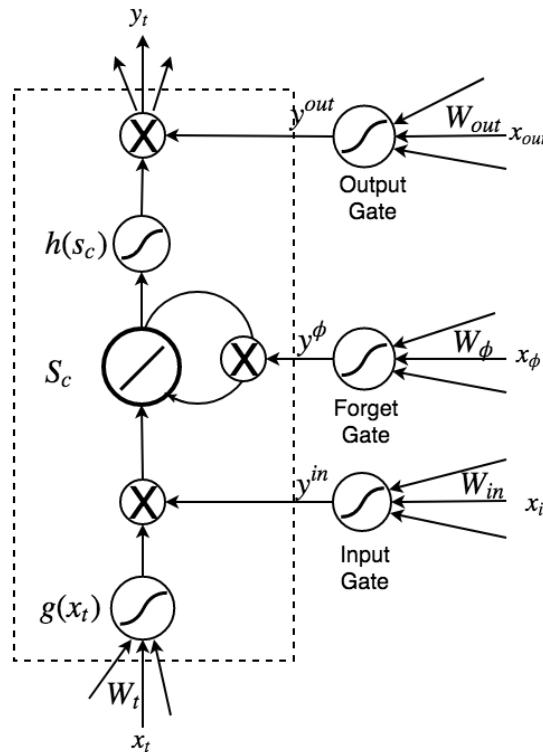
Vanilla Recurrent Neural Network

Redes Neuronales LSTM

Notación



- x : entrada
- o : objetivo
- S : estado
- U, V, W : matrices



Celda

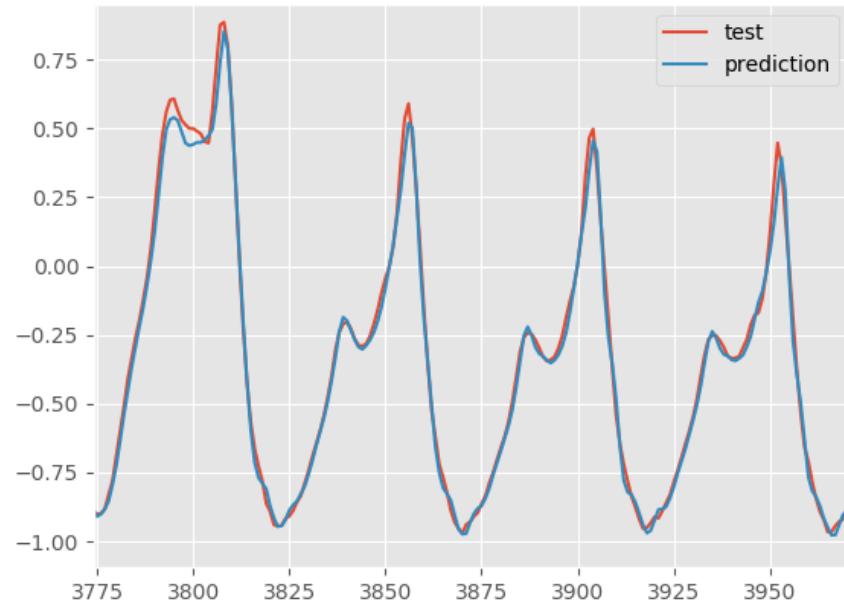
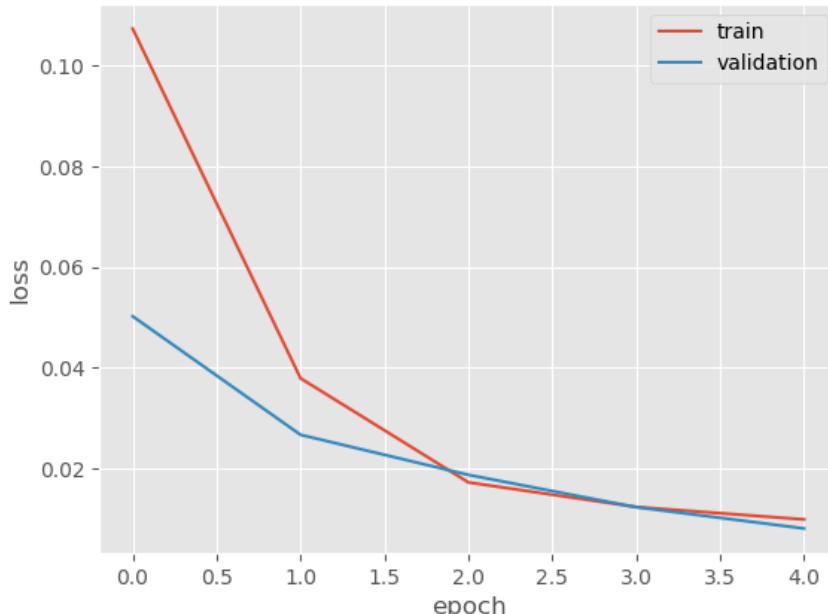
Contiene una entrada y una salida mas tres entradas de control:

- Input gate
- Forget gate
- Output gate

La función core es lineal

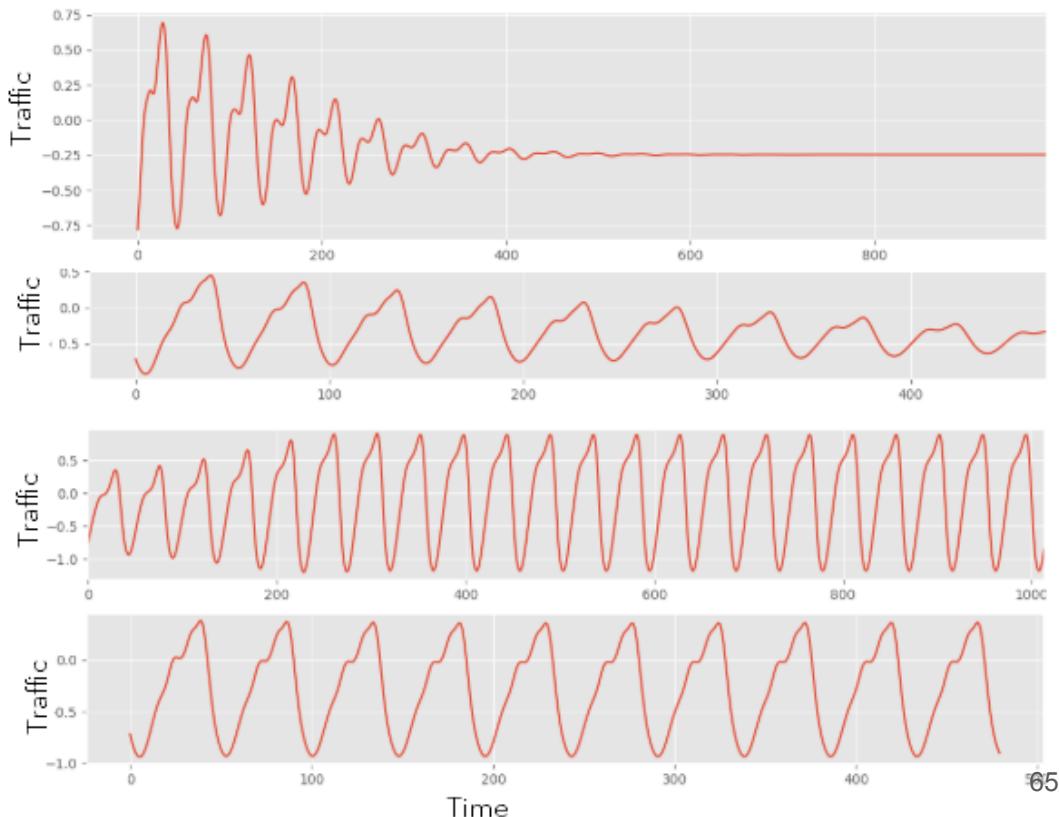
LSTM RNN: entrenamiento y test

Model train vs. validation loss



Predicciones usando ventana deslizante

- La técnica de ventana deslizante consiste en ir realimentando la serie predecida intervalo a intervalo.
- El largo de la ventana, la densidad de datos, la cantidad de períodos muestreados son factores de diseño.



Figuras de mérito

- Como figuras de mérito de los modelos elegimos algunas métricas que sirvan para hacer los modelos comparables.
- El error cuadrático medio, el máximo error absoluto, mínimo error porcentual (absoluto), entre otros.

Métricas de Error
Guía Rápida

	Selección de Modelo	Calibración	Toma de Decisiones
MAD			
ET			
RMSE			
MPE			
MAPE			
SFE			
BIAS			
GONA			

Resultados

Training set	ARIMA (p,d,q)	RMSE	MAE	MPE (%)	MAPE (%)	LSTM (inputs,batch, epochs)	RMSE	MAE	MAPE
CDN Google	1,2,1	13,08	10,35	1,16	2,40	144, 144, 5	8.7×10^{-4}	9.2×10^{-3}	< 0.00%
CND Netflix	1,2,1	27,70	19,76	0,22	2,86	144, 144, 5	3.3×10^{-3}	3.1×10^{-3}	< 0.00%
CDN Verizon	1,2,2	9,29	5,28	-1,72	20,63	336(7d), 336,5	2.5×10^{-4}	2.3×10^{-2}	< 0.0%
CDN Akamai	2,2,2	15,25	10,79	-1,54	10,79	336, 336,5	5.7×10^{-4}	6.2×10^{-2}	< 0.0%

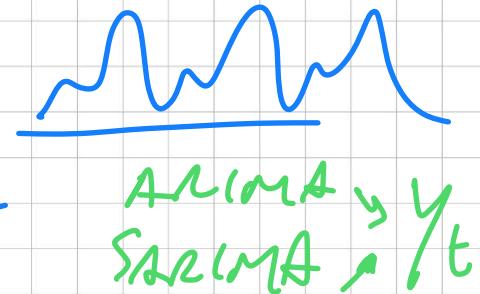
- Componentes estacionales requieren datos interanuales
- Para este dataset:
 - LSTM performance bien en el muy corto plazo
 - ARIMA ofrece tendencias explicables a largo plazo
- Compromiso entre cantidad de datos y método utilizado
- Posibles mejoras extendiendo el tamaño de la red LSTM

$$① Y_t = \mu_t + X_t$$

$f(\cdot)$

$$X_t = Y_t - \hat{\mu}_t$$

estacionario?



$$X_t = f(\mu_t)$$

si: ARMA
no: + preproc.

$$② Y_t = \sigma_t + X_t \Rightarrow Y_t \text{ SARIMA}$$

$$\sigma_t = \sigma_{t-1} + \epsilon_t$$