Análisis de Series de Tiempo

Carrera de Especialización en Inteligencia Artificial

Clase [4]

Ing. Magdalena Bouza, Ing. Carlos German Carreño Romano

Agenda

- Estimación del orden del modelo
- Modelos estacionales
- Estimación de parámetros
- Redes Neuronales para series de tiempo
- Revisión TP: primer entrega

Estimación del orden

Función de autocorrelación parcial

Define el efecto que tiene Y(t-k) sobre Y(t), descontando el efecto de las variables Yt-1,

$$\alpha(1) = corr(Y_{t+1}, Y_t), \text{ for } k = 1$$

$$\alpha(k) = corr(Y_t - (\beta_1 Y_{t-1} + \ldots \beta_{k-1} Y_{t-(k+1)}), \\ Y_{t-k} - (\beta_1 Y_{t-(k-1)} + \ldots + \beta_{k-1} Y_{t-1})) \text{ para } k \geq 2$$

$$\alpha(k) = corr((Y_t - P_{t,k}(Y_t), Y_{t-k} - P_{t,k}(Y_{t-K})), \text{ para } k \geq 2$$

$P_{t,k}(x)$ proyección ortogonal sobre las funciones lineales generadas por $Y_{t-1}, \ldots, Y_{t-(k-1)}$

Y_{t-2} Y_{t-1} Y_t

AR(1) $Y_t = a_1 Y_{t-1} + \varepsilon_t$

$\text{Cov}(Y_t, Y_{t-1}) = C_1 = \text{Cov}(a_1 Y_{t-1} + \varepsilon_t, Y_{t-1})$

$= a_1 \text{Var}(Y_{t-1})$

$\text{Cov}(Y_t, Y_{t-2}) = C_2 = a_1^2 \frac{\sigma^2}{1-a_1^2}$

$\frac{\sigma^2}{1-a_1^2}$

$\hat{Y}_t = \delta + a_1 Y_{t-1}$

ε_t

$\text{Cov}(Y_t - \hat{Y}_t, Y_{t-2} - \hat{Y}_{t-2}) = 0$

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

¿Cómo estimar el orden del modelo?

Model	PACF
White noise	The partial autocorrelation is 0 for all lags.
Autoregressive model	The partial autocorrelation for an AR(p) model is nonzero for lags less than or equal to p and 0 for lags greater than p.
Moving-average model	If $\phi_{1,1} > 0$, the partial autocorrelation oscillates to 0.
	If $\phi_{1,1} < 0$, the partial autocorrelation geometrically decays to 0.
Autoregressive–moving-average model	An ARMA(p, q) model's partial autocorrelation geometrically decays to 0 but only after lags greater than p.

Ref.: https://en.wikipedia.org/wiki/Partial_autocorrelation_function

¿Cómo estimar el orden del modelo?

Usando la función de autocorrelación y la función de autocorrelación parcial.

Para un modelo **MA(q)**

- Para k>q la **autocorrelación** es nula

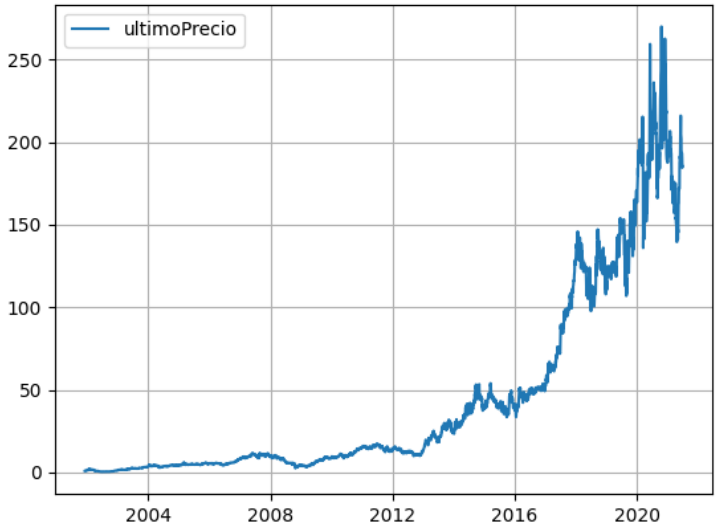
Para un modelo **AR(p)**

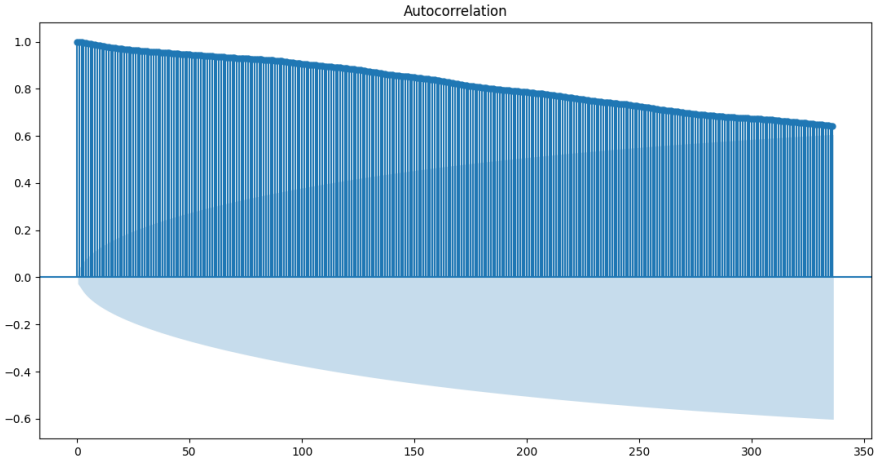
- Para k>p, la **autocorrelación parcial** es nula

Para un modelo **ARMA(p,q)**

- Para k>p la **autocorrelación parcial** decae a cero

Función de autocorrelación



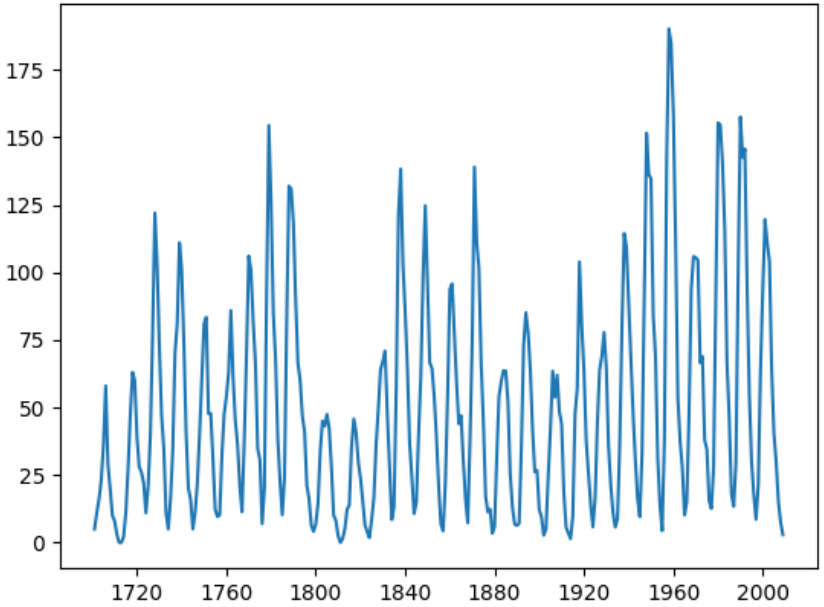


→ Indicio de que la serie es NO estacionaria

Función de autocorrelación

Ejemplo con dataset público

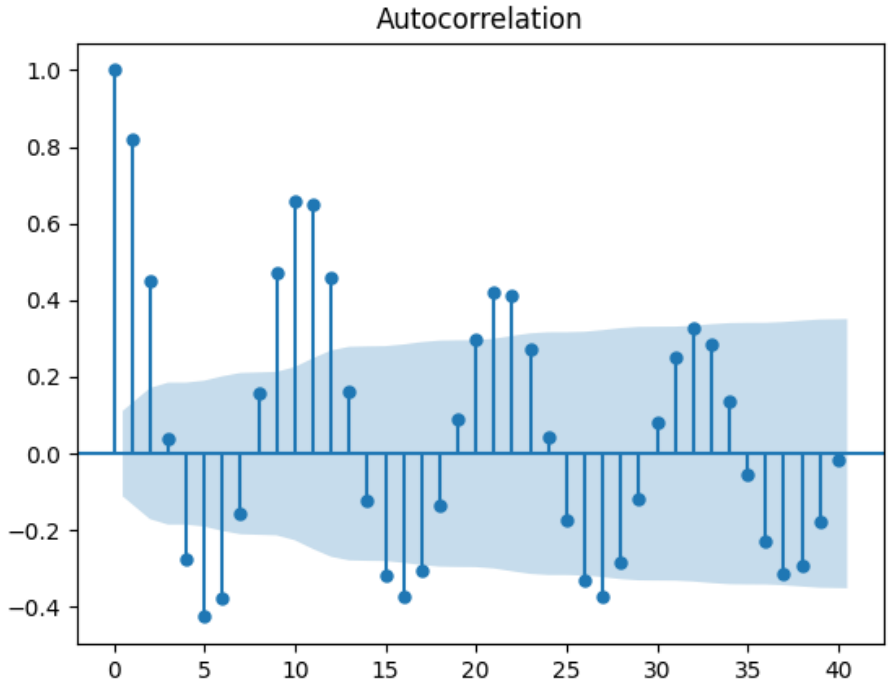
```
import statsmodels.api
as sm

dta =
sm.datasets.sunspots.loa
d_pandas().data
```



Función de autocorrelación

Ejemplo con dataset público

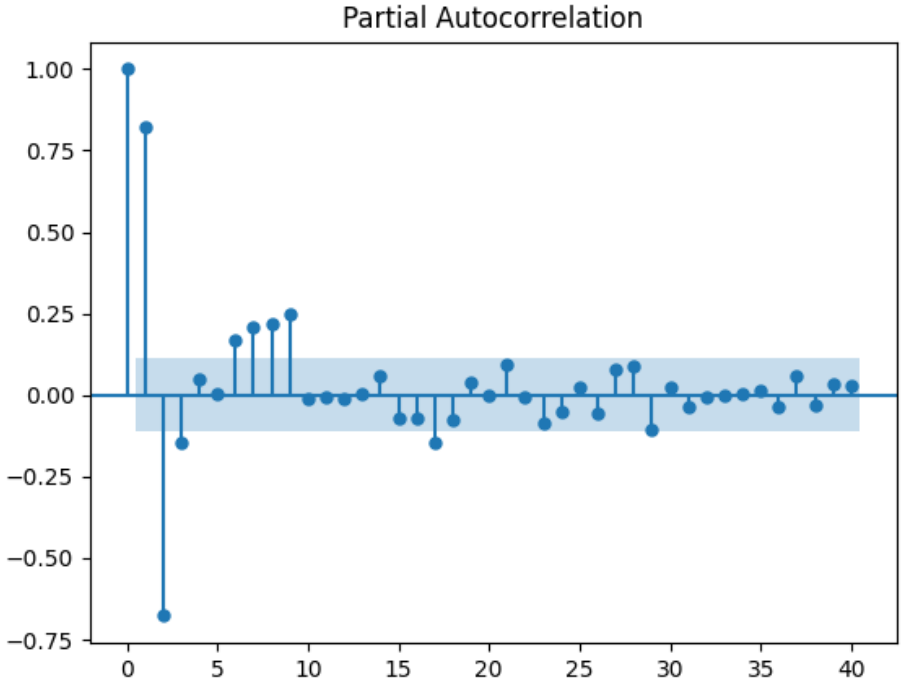
```
sm.graphics.tsa.plot_acf
(dta.values.squeeze(),
lags=40)
```



Función de autocorrelación parcial

Ejemplo con dataset público

```
sm.graphics.tsa.plot_pacf
(dta.values.squeeze(),
lags=40, method="ywm")
```



Sobre la estimación del orden de integración

¡Cuidado con no diferenciar de más!

La diferencia de cualquier serie estacionaria sigue siendo estacionaria, **pero**:

- Diferenciar más de lo necesario introduce correlaciones innecesarias en los datos, complicando el modelado.
- Sobrediferenciar lleva a modelos no invertibles (el mismo modelo se puede representar con distintas combinaciones de parámetros AR y MA)

Modelos estacionales

AR, MA usando Operadores

AR(p)

$$e_t = Y_t - \sum_{k=1}^{p} a_k Y_{t-k}$$

$$e_t = (1 - \sum_{k=1}^{p} a_k B^k) Y_t$$

$$e_t = \phi(B) Y_t$$

MA(q)

$$Y_t = e_t - \sum_{k=1}^{p} b_k e_{t-k}$$

$$Y_t = (1 - \sum_{k=1}^{p} b_k B^k) e_t$$

$$Y_t = \theta(B) e_t$$

Modelo SARIMA multiplicativo

In general, then, we define a **multiplicative seasonal ARMA$(p,q)\times(P,Q)_s$ model with seasonal period s** as a model with AR characteristic polynomial $\phi(x)\Phi(x)$ and MA characteristic polynomial $\theta(x)\Theta(x)$, where

$$\left.\begin{aligned}\phi(x) &= 1-\phi_1 x-\phi_2 x^2-\cdots-\phi_p x^p\\ \Phi(x) &= 1-\Phi_1 x^s-\Phi_2 x^{2s}-\cdots-\Phi_P x^{Ps}\end{aligned}\right\}$$

and

$$\left.\begin{aligned}\theta(x) &= 1-\theta_1 x-\theta_2 x^2-\cdots-\theta_q x^q\\ \Theta(x) &= 1-\Theta_1 x^s-\Theta_2 x^{2s}-\cdots-\Theta_Q x^{Qs}\end{aligned}\right\}$$

The model may also contain a constant term θ_0. Note once more that we have just a special ARMA model with AR order $p+Ps$ and MA order $q+Qs$, but the coefficients are not completely general, being determined by only $p+P+q+Q$ coefficients. If $s=12$, $p+P+q+Q$ will be considerably smaller than $p+Ps+q+Qs$ and will allow a much more parsimonious model.

Notación equivalente a(B) con \phi(B), b(B) con \theta(B). Usamos \phi y \theta para evitar notar B(B).

Modelo SARIMA multiplicativo

ARMA(p,q)

$$\phi(B)Y_t = \theta(B)e_t$$

ARIMA(p,d,q)

$$\phi(B)(1-B)^d Y_t = \theta(B)e_t$$

SARMA(p,q)(P,Q)$_s$

$$\phi(B)\Phi(B^s)Y_t = \theta(B)\Theta(B^s)e_t$$

Recordar que $\nabla_s^d Y_t = (1-B_s)^d Y_t$

SARIMA(p,d,q)(P,D,Q)$_s$

$$\phi(B)\Phi(B^s)((1-B)^d(1-B^s)^D Y_t) = \theta(B)\Theta(B^s)e_t$$

Modelo SARIMA multiplicativo

SAR(p)(P)s como AR(p) multiplicativo con AR(P)s del polinomio de la componente estacional

$$\phi(B)\Phi(B^s)Y_t = e_t$$

SMA(q)(Q)s como MA(q) multiplicativo con MA(Q)s del polinomio de la componente estacional

$$\theta(B)\Theta(B^s)e_t = Y_t$$

ARIMA como ARMA diferenciado

Si

$$W_t = \nabla^d Y_t$$

es ARMA, entonces Y_t es ARIMA.

SARIMA como SARMA diferenciado

Si

$$W_t = \nabla^d \nabla_s^D Y_t$$

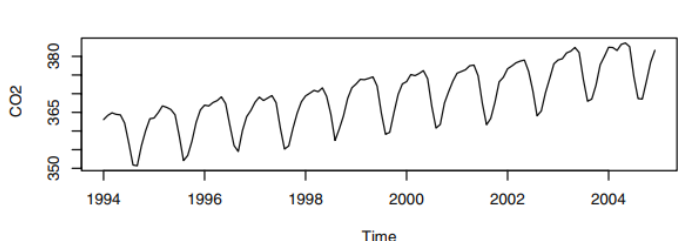
es SARMA, entonces Y_t es SARIMA

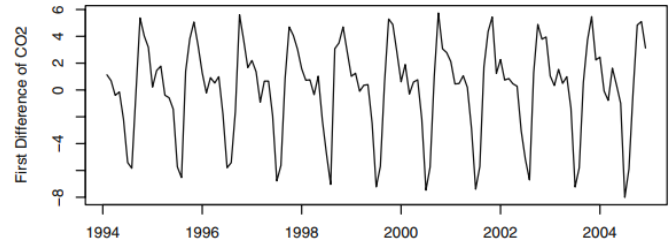
Especificación del modelo

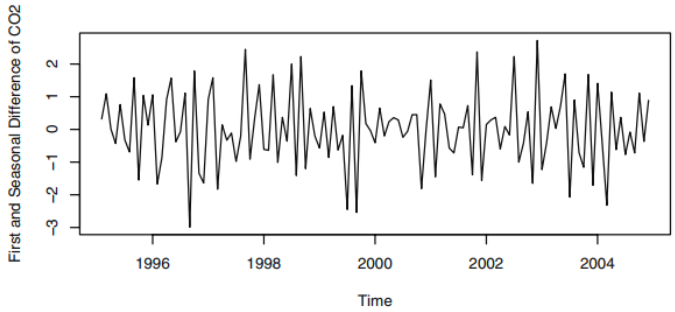
Se usan las mismas ideas introducidas para modelos ARMA. y ARIMA.

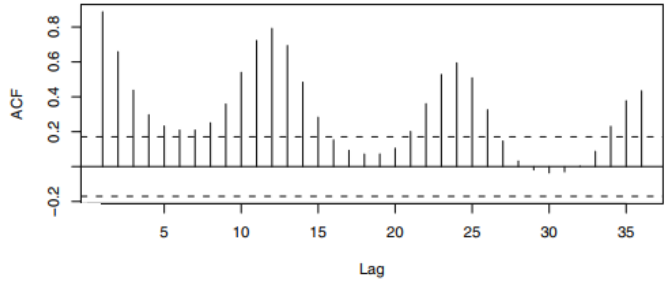
1) Inspeccionar la serie de tiempo y su función de autocorrelación muestral. ¿Observo alguna tendencia y/o algún comportamiento estacional?
2) Proponer transformaciones (ej. diferenciar) y volver a analizar la serie resultante

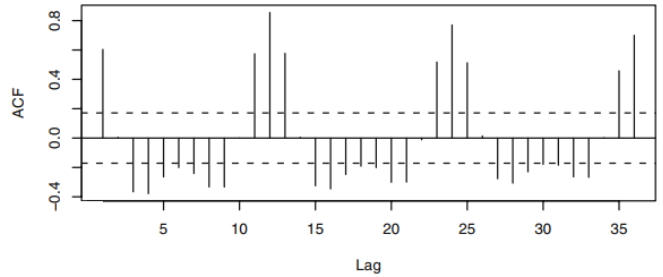
Especificación del modelo - ejemplo

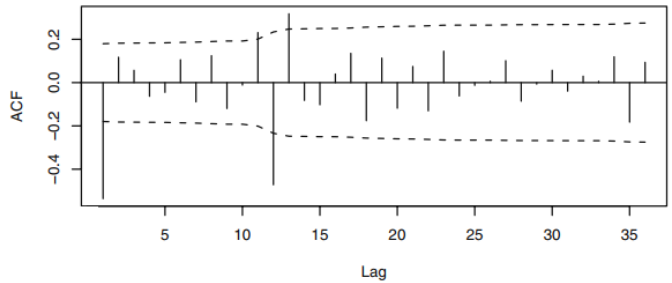












Estimación de parámetros

¿Cómo estimar los parámetros de un modelo?

En general no conocemos los valores de $a_1, \ldots a_p$ y $b_1, \ldots b_q$ 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 $a_1, \ldots, a_p$:

$$\begin{cases} \hat{R}_1 = a_1 + a_2 \hat{R}_1 + \ldots + a_p \hat{R}_{p-1} \\ \quad \vdots \\ \hat{R}_p = a_1 \hat{R}_{p-1} + a_2 \hat{R}_{p-2} + \ldots + a_p \hat{R}_p \end{cases}$$

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 $\mathrm{AIC}_0 = N(\log 2\pi\hat{\sigma}_0^2 + 1) + 2$
2. For $m = 1, \cdots, M$, repeat the following steps

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

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

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

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

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

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

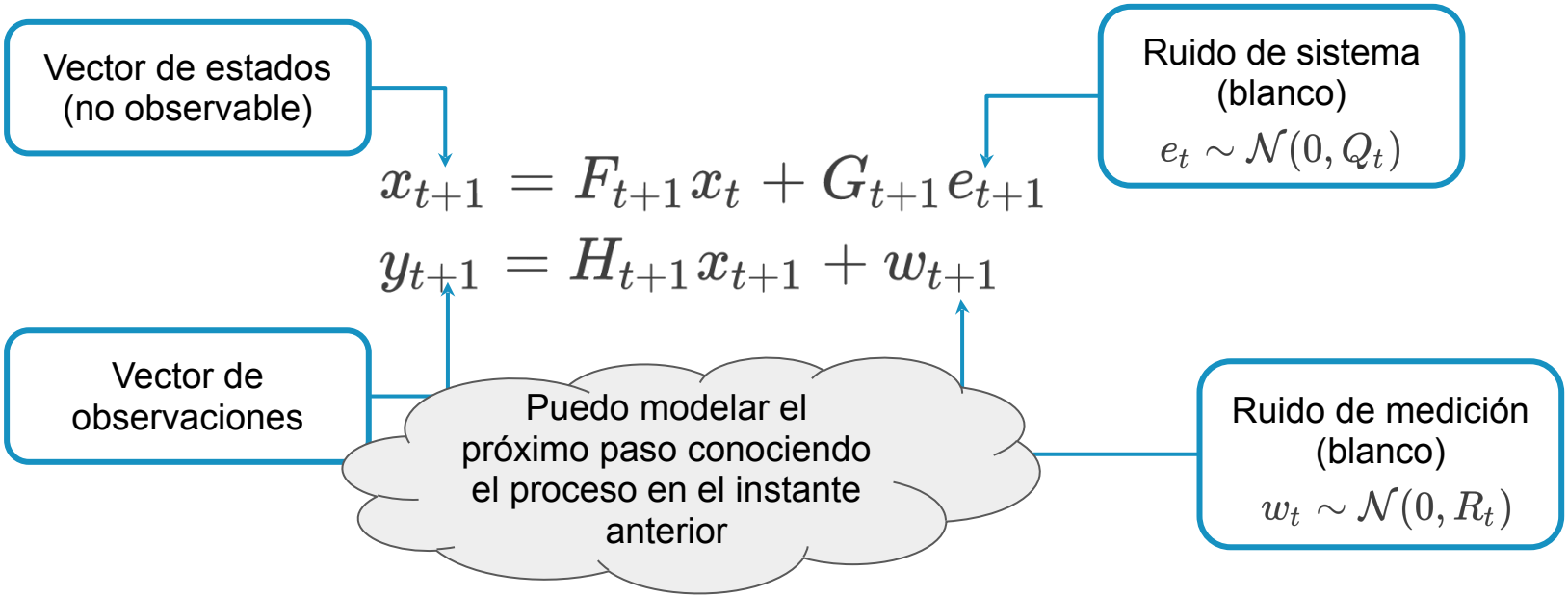
Hallar el EMV puede ser costoso computacionalmente pues hay que calcular Σ^{-1}

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?

Modelo de estados

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



Modelo de estados - Filtro de Karman

¿Cómo puedo estimar el siguiente paso?

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

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

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

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

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

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

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

Modelo AR como modelo de estados

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

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

Además, considerando $H = [1 \quad 0 \quad \ldots \quad 0]$, 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 $x_t = [y_t, y_{t-1}, \ldots, y_{t-p+1}]^T$ al igual que en el modelo AR. Las matric F y H también va a ser las mismas:

$$F = \begin{bmatrix} a_1 & a_2 & \ldots & a_p \\ 1 & 0 & \ldots & 0 \\ & \ddots & & \\ & & 1 & 0 \end{bmatrix} \quad H = [1\,0\,\ldots\,0]$$

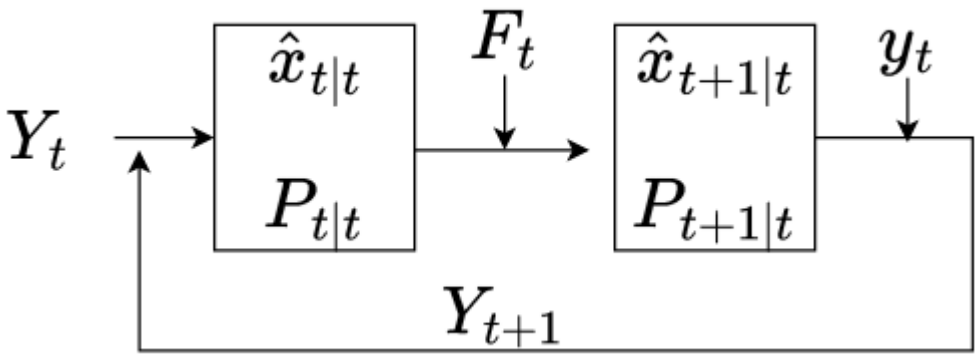
Lo que va a cambiar en el modelo ARMA es la matriz G, que debe incluir la dependencia con ruidos anteriores:

$$G = \begin{bmatrix} 1 & b_1 & \ldots & b_q \\ 0 & 0 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & 0 \end{bmatrix}$$

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.

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



¿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

$$y_t | Y_{t-1} \sim \mathcal{N}(y_{t|t-1}, d_{t|t-1})$$

donde

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

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,...,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.

Fuente: http://www.kris-nimark.net/pdf/Handout_S4.pdf

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

Redes Neuronales para Series de Tiempo

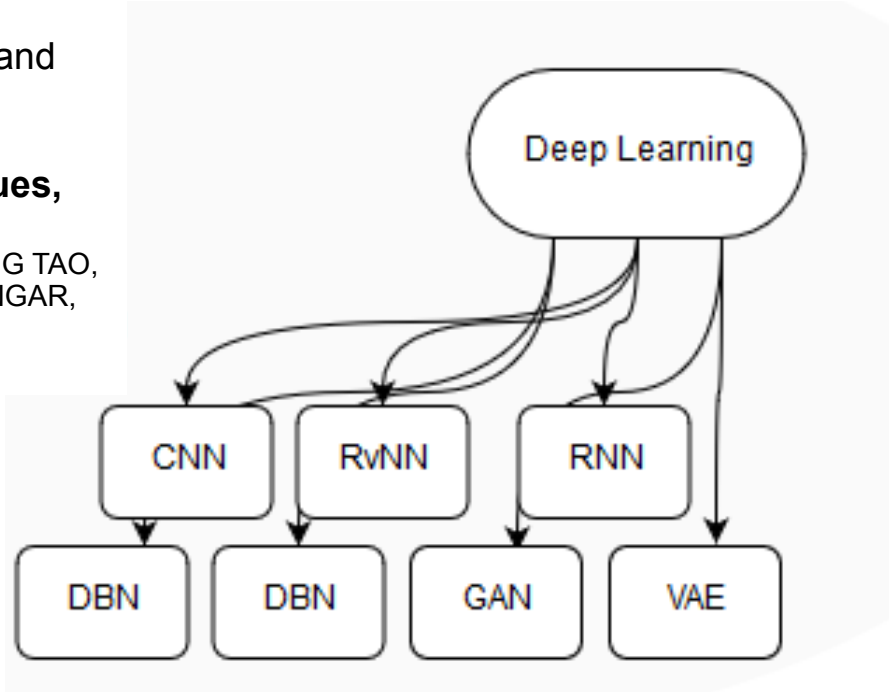
Redes Neuronales

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

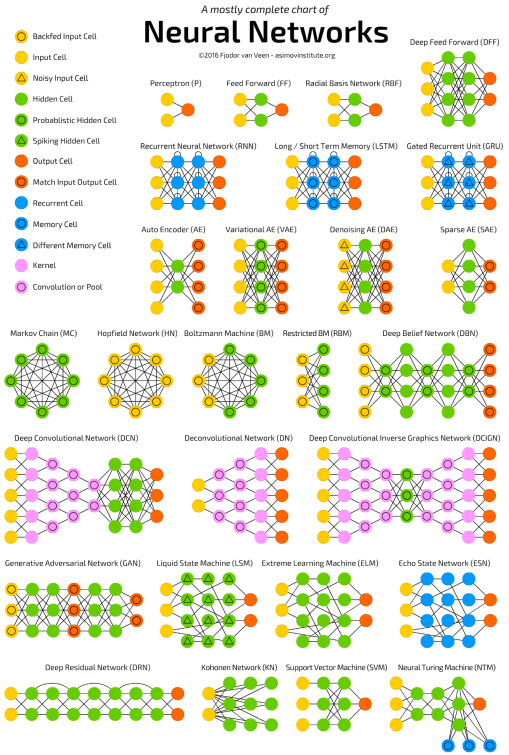
- Caffe, DL4j, Torch, Neon, Theano, MXNet, TensorFlow, CNTK

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 **X** such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}$. 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

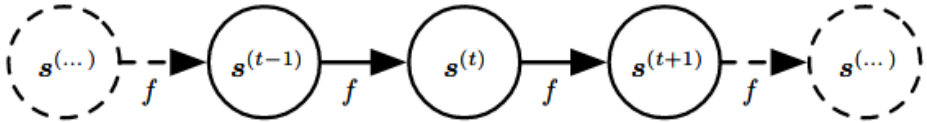
Redes Neuronales Recurrentes

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

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

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

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



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

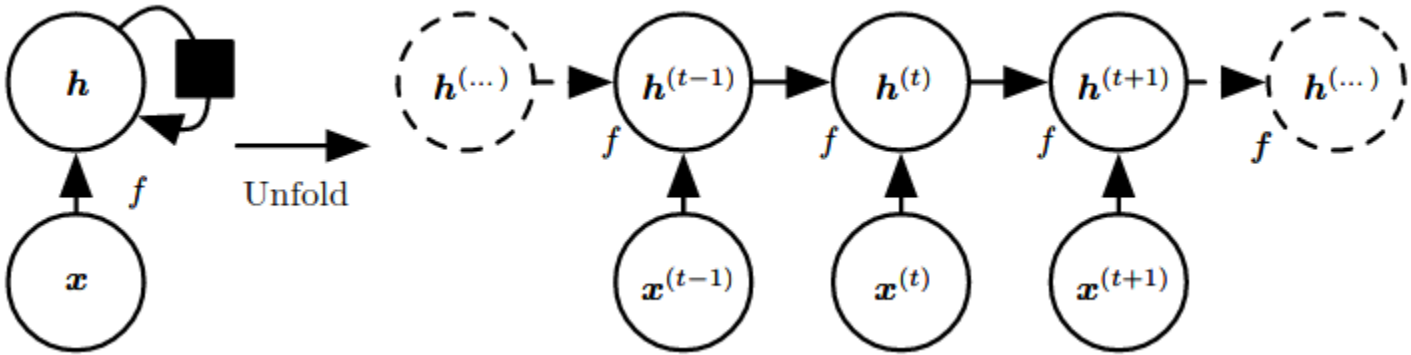
Ref.: Yoshua Bengio, Deep Learning Adaptive Computation and Machine Learning

Redes Neuronales Recurrentes

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

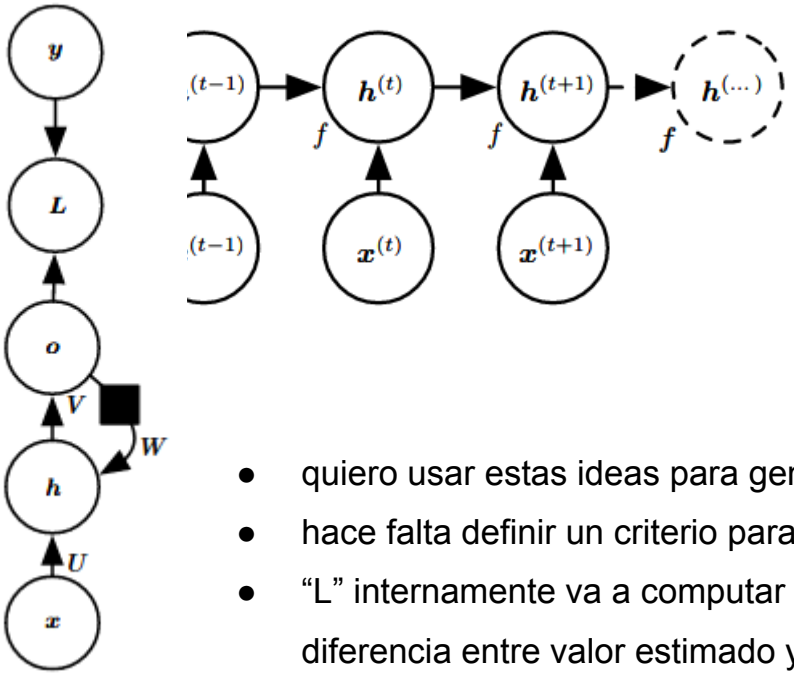
$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).\end{aligned}$$

- Usamos 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)



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

Redes Neuronales Recurrentes

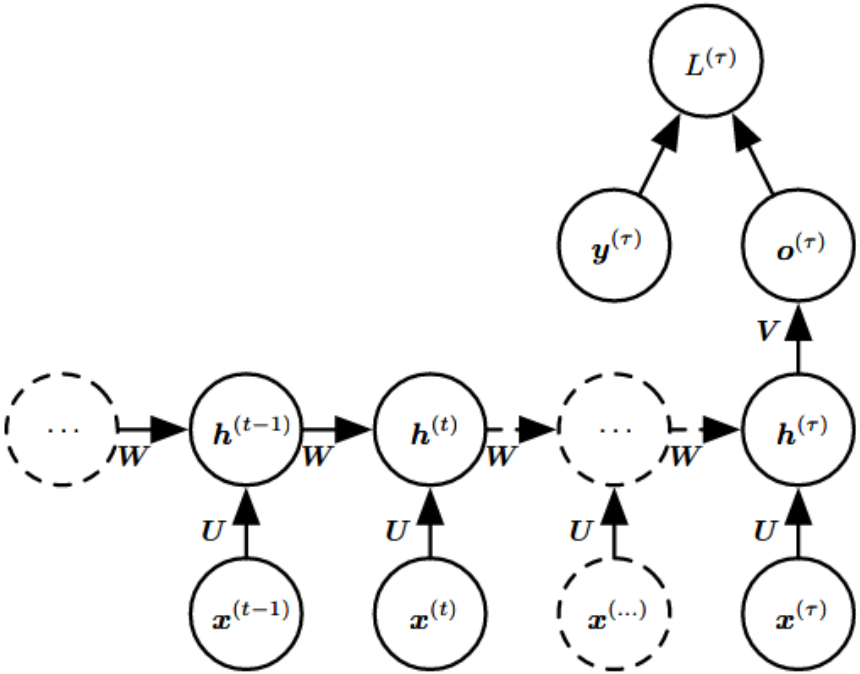


- hay una ventaja en esta notación que refiere al largo de la red: no es el largo de la serie, sino de transiciones de estados.
- es posible usar la misma función de transición f con los mismos parámetros en cada paso. Se posibilita **generalización**.

- quiero usar estas ideas para generar una salida o^(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^(t)) para y^(t) y va a buscar minimizar la diferencia entre valor estimado y valor obtenido.

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

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)

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

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)

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

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

***Ref**.: https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/*

Revisión TP: primer entrega