UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE CIENCIA

Departamento de Matemática y Ciencia de la Computación

Métodos de Aprendizaje en línea para Modelos ARIMA

Matías Cáceres Parravicini

Profesor guía: Felipe Elorrieta López

Trabajo de titulación presentado a la Facultad de Ciencia en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Estadístico

Santiago, Chile

2020

Agradecimientos

A mi familia por darme todo lo posible con mucho sacrificio y amor, para enfrentarme a este proceso profesional y culminar mi carrera.

A mi profesor guía Felipe Elorrieta, por aceptarme como tesista, por facilitarme un tema relacionado al área Series de Tiempo, por su gran paciencia y dedicación durante todo el proceso.

A las profesoras y profesores por darme herramientas y consejos útiles para este proceso.

A mis compañeros y compañeras de Ingesta, por toda la paciencia durante todos estos años de formación, por aquellas tardes de estudios con intenciones de superarse cada día y las veces que se compartió una gran amistad, los quiero.

A mis amigos y amigas de toda la vida, que con paciencia y cariño estuvieron apoyandome, los amo.

Resumen

En la presente tesis se describen los distintos modelos estacionarios de series de tiempo, el modelo no estacionario ARIMA, todos los elementos y caracteristicas que componen a estos modelos.

Posteriormente se describe el significado de los procesos batch, sus principales virtudes y defectos, en comparación a los procesos en línea.

Despues se explica la metodologia del aprendizaje en línea para la aplicación en series de tiempo, desde la entrada de los datos hasta el error de predicción. Todo esto ultimó, mediante los aspectos teoricos proveniente del algoritmo Newton-Step.

Se explica la manera de generar las distintas series de tiempo, con sus respectivos tamaños, repeticiones, parámetros y distribuciones de los términos de ruido.

Una ves realizadas estas simulaciones, se efectua la aplicación del método aprendizaje en línea y el método batch, obteniendo sus errores de predicción y así, poder compararlas en los distinto escenarios.

Palabras claves: Modelo ARIMA, procesos batch, métodos de aprendizaje en línea, algoritmo Newton-Step, error de predicción.

Índice de imágenes y gráficos

Índice general

Capítulo 1

Introducción

Apreciamos hoy en día que, en las últimas décadas, se ha generado una revolución de datos de manera masiva y continua, por lo tanto, ha sido necesario contar con la ayuda de avances tecnológicos; esto, con el propósito de retener, almacenar y/o procesar estás grandes cantidades de datos. Actualmente, toda esta información que se recopila constantemente es denominada "Data Stream".

En este contexto, uno de los principales problemas y desafíos cuando se trabaja a grandes escalas de datos, ocurre al volverse costosos los procesos, en términos de tiempo. Dentro de los últimos años, se han investigado y aplicado varios algoritmos de aprendizaje en línea efectivos para la variedad de datos, con el fin de abordar esta última necesidad del costo computacional. Entre los algoritmos se presentan: la teoría de juegos, la teoría de la información, el aprendizaje automático (Bubeck, 2011) y la minería de datos. La mayoría de los algoritmos en línea, propuestos anteriormente, entran como marco de referencia a la optimización convexa en línea (Zinkevich, 2003).

En la actualidad, los datos se están recibiendo a cada segundo en diversos ámbitos, generando análisis, predicciones y/o clasificaciones. Es por esto, que los mode-

los estadísticos requieren una optimización en línea, provenientes de algoritmos de aprendizaje. Un ejemplo dentro de los modelos estadísticos más utilizados, por su popularidad en el pronóstico de series de tiempo, es el Autorregresivo Integrado de Media Móvil (ARIMA, en inglés), el cual cuenta con propiedades estadísticas que poseen un carácter flexible. A pesar de esto, la gran mayoría de algoritmos, no cuentan con una técnica o método de aprendizaje implementadas en los flujos de datos continuos, para poder evitar, por ejemplo, la falta de adaptación ante un cambio en los datos.

Por esta razón, se utilizarán datos simulados, los cuales provienen de diversos procesos de series de tiempo ARIMA con dos técnicas, reconocidas en el mundo del aprendizaje en línea, y de está manera posibilitar adaptarse a la potencialidad de datos continuos.

1.1. Contextualización del problema

En estos tiempos, las series de tiempo han jugado un papel importante, en una amplia gama de áreas que incluyen el análisis del habla (Rabiner and R, 2011), cancelación del ruido (Gao J and W, 2010), el análisis del mercado financiero Brockwell and R (2009), datos astronómicos (Elorrieta, 2018), etc... Por lo general, estos modelos pueden recoger las observaciones del pasado y descubrir su relación subyacente.

Entre los modelos más usados en el análisis de series de tiempo, está el Autorregresivo de Media Móvil (ARMA, en inglés) (Hamilton, 1994), originándose a partir del modelo Autorregresivo (AR, en inglés) y el Modelo de Media Móvil (MA, en inglés). El proceso ARMA es estacionario, además describe comportamientos lineales ruidosos y a su vez representan diferentes tipos de Series de Tiempo, debido a su

capacidad flexible.

Lamentablemente, los procesos ARMA no sirven ante una situación de series de tiempo no estacionarias, es decir, se necesitan utilizar técnicas de diferenciación, para eliminar las componentes que generan la no-estacionaridad. De este modo, llega a los procesos que se utilizarán en esta pluralidad de esta tesis. Los procesos ARIMA, son uno de los más conocidos cuando no se presenta estacionaridad, que se origina a partir de un AR, MA y el parámetro de integración.

Sin embargo, estos procesos ARIMA como otros modelos estadísticos, utilizan criterios de información como: Los criterios de bondad de ajuste, el criterio de información Akaike (AIC, en inglés) (Akaike, 1998) y el criterio de información Bayesiano (BIC, en inglés) (Schwarz, 1978); o los métodos de estimación de los parámetros, tales como mínimos cuadrados, métodos de máxima verosimilitud o para los procesos AR, el estimador de momentos Yule-Walker. En resumen, estos últimos mencionados, se realizan mediante procesamientos en Batch, es decir, procesos en lotes que consumen mucha memoria por los grandes volúmenes de datos en un mismo instante de tiempo, alcanzando una velocidad más lenta (Hoi S C and P, 2014).

Por lo tanto, la principal noción de esta tesis u la idea principal es aplicar diferentes algoritmos mediante procesamientos en Stream, es decir, analizar los datos continuamente, evitando procesar grandes cantidades de datos, en comparación a los procesos Batch. Con la finalidad de hacer frente al modelamiento de series de tiempo en tiempo real (Anava O and Shamir, 2013) reformulando en una optimización online (sin términos de ruido aleatorio) (Chenghao L and S, 2016).

Entre los diferentes algoritmos para los procesos ARIMA, se propone el siguiente método:

- Online Newton Step (ONS) (Hazan E and S, 2007): Análogo al método de Newton-Raphson. Algoritmo que iterativamente optimiza los vectores de coeficientes del modelo, mediante una simple modificación del punto elegido en la iteración anterior.

Con el objetivo de estimar los parámetros de forma eficiente y escalable, garantizando al método Online ARIMA que los algoritmos propuestos se aproximarán asintóticamente al rendimiento del mejor modelo ARIMA(Anava O and Shamir, 2013).

1.2. Objetivos

1.2.1. Objetivo general

- Implementar un método de aprendizaje en línea para modelos ARIMA.

1.2.2. Objetivos específicos

- Implementar el algoritmo Newton-Step, orientado a Procesos ARIMA y sus derivados, en el software estadístico R.
- Obtener las estimaciones, de los parámetros del método propuesto, para comparar con el método en lote.
- Evaluar el rendimiento del método, mediante su error de predicción, en diferentes contextos y diversos datos simulados.

Capítulo 2

Metodología

En primera instancia, se realizará una extensa revisión bibliográfica, sobre los algoritmos de aprendizaje en línea, implementados en series de tiempo, con las comparaciones teóricas y computacionales, entre los algoritmos propuestos, con el fin de desarrollar de forma optima este proyecto.

Luego, se implementará los algoritmos en el software R (Team, 2020). Para ello, se indagará en la librería que posee para el análisis de series de tiempo, "astsa"(Stoffer, 2020).

Posteriormente, se procederá a evaluar el rendimiento de los métodos, mediante el error cuadrático medio en línea (RMSE), puesto que se pretende comparar lo programado para datos en línea, frente al método convencional. También se quiere estudiar la convergencia de los parámetros y cuánto tarda en llegar al verdadero valor. Se harán simulaciones y se verificará como es el proceso de estimación online.

Finalmente, se procederá a usar la aplicación en datos simulados para ver su rendimiento en la predicción.

Capítulo 3

Marco teórico

3.1. Preliminares y Modelo

3.1.1. Series de Tiempo

Una Serie de Tiempo es una secuencia de valores observados, provenientes de datos cuantitativos que se miden en intervalos de tiempo sucesivos, es decir, espaciados uniformemente entre las observaciones. Estas secuencias, pueden tomar diferentes comportamientos, tales como tendencia lineal, cuadrática, variaciones cíclicas, sinusoidales o simplemente algún patrón no conocido. En términos generales, asumiremos que el tiempo t es una variable discreta estrictamente creciente, tal que X_t se denota como la observación en el instante t.

Por otra parte, para poder hablar sobre los modelos comúnmente usados en Series de Tiempo, es importante definir algunos conceptos:

3.1.1.1. Procesos Estacionarios:

Los Procesos Estacionarios son importantes, debido a su facilidad de análisis y su gran ayuda para la toma de decisiones. Estos procesos se dividen en 2 tipos:

- Procesos Estrictamente Estacionarios o Estacionaridad fuerte: Un proceso x_t es estricto, cuando todas las distribuciones de dimensión finita k, son iguales en el tiempo, es decir:
  $F(X_1, ..., X_k) = F(X_{t+1}, ..., X_{t+k}); \forall t, k \in \mathbb{Z}$

- Procesos Débilmente Estacionarios: Un Proceso es Estacionario de segundo orden:

  1. Tiene media constante, es decir, $\mathbb{E}(X_t) = \mu < +\infty, \forall t \in \mathbb{Z}$
  2. Tiene varianza constante, es decir, $\mathbb{V}(X_t) = \sigma^2 < +\infty, \forall t \in \mathbb{Z}$
  3. Tiene una función de autocovarianza tal que, $\mathbb{C}ov(X_t, X_{t\pm k}) = \gamma(k), \forall t, k \in \mathbb{Z}$, es decir, el valor de covarianza dependa solamente del tiempo que paso entre ambos periodos.

3.1.1.2. Ruido Blanco:

Un caso particular de un Proceso Débilmente Estacionario, es el Ruido Blanco aleatorio de las Series de Tiempo $\{\epsilon_t\}$. Están presentes en todo análisis y usualmente se utilizan con las siguientes propiedades:

1. $\mathbb{E}(\epsilon_t) = \mu < +\infty, \forall t \in \mathbb{Z}$
2. $\mathbb{V}(\epsilon_t) = \sigma^2 < +\infty, \forall t \in \mathbb{Z}$
3. $\mathbb{C}ov(\epsilon_t, \epsilon_{t\pm k}) = 0, \forall t, k \in \mathbb{Z}$

Comúnmente la media constante se toma como $\mathbb{E}(\epsilon_t) = 0$, obteniendo la notación que se ocupará para el ruido blanco centrado aleatorio, $\epsilon_t \overset{iid}{\sim} RB(0, \sigma^2)$.

3.1.2. Modelos ARIMA

Para poder hablar sobre los modelos estadísticos de Series de Tiempo, tenemos que diferenciar dos tipos:

3.1.2.1. Modelos Estacionarios:

Estos procesos se encuentran en la naturaleza estacionaria, sin necesidad de realizar alguna transformación matemática, para la facilidad de análisis, teniendo esto en cuenta se distinguen tres procesos:

1. Modelo Autorregresivo: Se dice modelo Autorregresivo de orden p al proceso denotado como AR(p), si satisface:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \epsilon_t, \tag{3.1}$$

Donde $\epsilon_t \overset{iid}{\sim} RB(0, \sigma^2)$ y los coeficientes $\alpha_1, ..., \alpha_p$ son fijos.
De la misma manera, se puede reescribir en una forma más compacta:

$$\Phi_p(B)X_t = \epsilon_t, \tag{3.2}$$

De modo que B es operador de rezago y $\Phi()$ es el polinomio característico definido como $\Phi_p(B) = (1 - \alpha_1 B - \alpha_2 B^2 - ... - \alpha_p B^p)$. Además, se debe tener en cuenta que las raíces del polinomio Autorregresivo deben estar fuera del círculo unitario, para cumplir que sea un proceso causal, es decir, la condición de estacionaridad, y así equivalente a un $MA(\infty)$.

2. Modelo De Medias Móviles: Sea el Modelo De Medias Móviles de orden q denotado como MA(q), si satisface:

$$X_t = \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t, \tag{3.3}$$

Por esto $\epsilon_t \overset{iid}{\sim} RB(0, \sigma^2)$ y los coeficientes $\beta_1, ..., \beta_q$ son fijos.
De la misma manera, se puede reescribir en una forma más compacta:

$$X_t = \Theta_q(B)\epsilon_t, \tag{3.4}$$

Por lo que $\Theta()$ es el polinomio característico definido como $\Theta_p(B) = (1-\beta_1 B-\beta_2 B^2-...-\beta_p B^p)$. Además, se debe tener en cuenta que las raíces del polinomio de Medias Móviles, deben estar fuera del círculo unitario para cumplir que sea un proceso invertible, es decir, equivalente a un $AR(\infty)$.

3. Modelo Autorregresivo de Medias Móviles: Este proceso se denota como Modelo ARMA(p,q). Por consiguiente, el orden p pertenece a lo Autorregresivo y q a la de Medias Móviles, esto es, el Proceso ARMA(p,q) es la combinación de AR(p) y MA(q), satisfaciendo:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t, \tag{3.5}$$

De este modo $\epsilon_t \overset{iid}{\sim} RB(0, \sigma^2)$, y los coeficientes $\alpha_1, ..., \alpha_p$ y $\beta_1, ..., \beta_q$ son fijos. por lo tanto, se puede reescribir en una forma más compacta:

$$\Phi_p(B)X_t = \Theta_q(B)\epsilon_t, \tag{3.6}$$

Vale decir que $\Phi()$ es el Polinomio Autorregresivo y $\Theta()$ es el Polinomio De Medias Móviles. Además, se debe tener en cuenta que las raíces del Polinomio Autorregresivo, deben estar fuera del círculo unitario, para cumplir que sea un proceso causal, es decir, la condición de estacionaridad. Así pues, se debe tener en cuenta para el análisis, que el polinomio $\Theta()$ debe ser invertible.

3.1.2.2. Modelos No Estacionarios:

Al encontrarse con Series de Tiempo, usualmente estos no son procesos estacionarios, por lo tanto, se debe utilizar el Método De Diferenciación, con el objetivo de realizar análisis sobre procesos estacionarios. Este método es útil, debido a la transformación de una serie no estacionaria. Por ejemplo, en una serie con tendencia determinista se le aplica este método, obteniendo como resultado una serie estacionaria.

En el siguiente ejemplo del método, se tiene el cálculo del primer orden de diferenciación de X_t, obteniendo $\bigtriangledown X_t = (1-B)X_t = X_t - X_{t-1}$, el segundo orden de X_t es $\bigtriangledown^2 X_t = (1-B)^2 X_t = \bigtriangledown X_t - \bigtriangledown X_{t-1}$ y así sucesivamente las diferenciaciones de X_t.

Debido al Método De Diferenciación, podemos hablar sobre el modelo Autorregresivo Integrado de Medias Móviles, denotado como ARIMA(p,d,q), conteniendo éste los procesos AR(p), MA(q) y por último d, que corresponde a la cantidad de veces que fue diferenciada la serie.

En términos generales, si $X_t \sim ARIMA(p,d,q)$ entonces, $\bigtriangledown^d X_t \sim ARMA(p,q)$, por lo tanto, X_t satisface:

$$\bigtriangledown^d X_t = \sum_{i=1}^{p} \alpha_i \bigtriangledown^d X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t, \tag{3.7}$$

Por lo que $\epsilon_t \overset{iid}{\sim} RB(0,\sigma^2)$, d es fijo y los coeficientes $\alpha_1, ..., \alpha_p$ y $\beta_1, ..., \beta_q$ son fijos.
De la misma manera, se puede reescribir este proceso en una forma más compacta:

$$\Phi_p(B)(1-B)^d X_t = \Theta_q(B)\epsilon_t, \tag{3.8}$$

De modo que $\Phi()$ es el Polinomio Autorregresivo, $\Theta()$ es el Polinomio De Medias Móviles y $(1-B)^d$ representa las d veces que fue diferenciada la serie. Además, se debe tener en cuenta que las raíces del Polinomio Autorregresivo, deben estar fuera del círculo unitario, para cumplir que el proceso $\bigtriangledown^d X_t$ sea causal, y así cumpla la condición de estacionaridad.

El pronóstico con ARIMA, denota una reversión del proceso diferencial, es decir, supone que X_t satisface un modelo ARIMA(p,d,q) y así podemos predecir el orden diferencial d de la observación del tiempo t+1 como $\bigtriangledown^d \tilde{X}_{t+1}$ y predecir la observación

del tiempo t+1 como:

$$\tilde{X}_t = \bigtriangledown^d \tilde{X}_t + \sum_{i=0}^{d-1} \bigtriangledown^i X_{i-1} \tag{3.9}$$

3.2. Procesos Batch

Los procesos Batch o por lote, son sistemas de productos, destinados a las fábricas, que posteriormente se separan en lotes a medida que avanzan los procesos de producción. Este programa tiene como principal característica, la ejecución de una interacción con el usuario, sin control o supervisión.

Normalmente, este tipo de procesos se aplican en trabajos con grandes cantidades de datos, porque una posible aplicación manual en este tipo de sistema cargado de información, puede resultar tedioso, por lo tanto, y propenso a errores. Por consiguiente, se recomienda dividir los lotes, para que de esta manera se pueda evitar la acumulación de la información. Lo ideal es trabajar con flujos de información de una sola pieza, es decir, sin acumulamiento de los datos, sin embargo, esta última corriente no pertenece a los procesos Batch, sino a los Streaming Data.

Se debe agregar, por lo tanto, que el sistema de procesamiento por lotes es recomendable cuando la demanda de análisis o acumulación de información no es periódica, continua o suficientemente extensa como para poner en marcha el proceso en un solo lote. En este último caso, se lograría un tamaño óptimo para los lotes y así, poder evitar la acumulación en grandes cantidades de datos, y evitar también que ciclos de tiempo de los procesos sean largos y costosos. Ahora, si se habla de cambios dentro de la información, los procesos Batch se vuelven tediosos y largos, para el procesamiento de los datos; esto se debe a que el sistema es poco flexible y el cambio de modelos implique mayor dificultad.

Teniendo en cuenta estas características de los procesos Batch mencionadas anteriormente y las grandes innovaciones digitales de hoy en día, se inicia la competencia para evitar los problemas que genera el sistema por lotes, los mencionados anteriormente “Streaming Data”

Los Streaming Data o transmisión de datos, son procesos que dan la facilidad de realizar análisis continuos de la información, es decir, realizan análisis en pequeñas cantidades, para evitar la acumulación de datos, y de esta manera, se adaptan a la realidad. inmersa en avances tecnológicos y demandas de la nueva era del Big Data.

Los Streaming Data, también reúnen el análisis de la información, sin importar el cambio de velocidad de los datos, los que pueden separarse y recombinarse principalmente por flexibilidad y variedad, siendo esta última característica inoperante con los procesos Batch. Sin embargo, lo más importante, es que este tipo de sistema, puede integrar datos de diferentes fuentes de información.

Estos procesamientos continuos trabajan correctamente, a causa de una consecuencia de la conectividad de los datos, la interoperabilidad, que es la capacidad de un producto o sistema para trabajar con otros productos o sistemas. Además, también es posible el análisis de información en tiempo real, para conocer el comportamiento de los clientes o usuarios de una empresa. Los Streaming Data entregan óptimos análisis para el Big Data en tiempo real, reduciendo el tiempo de entrega de los paquetes de análisis de información.

Finalmente, lo más importante para diferenciar Los Streaming Data de los Proceso Batch, es la programación, porque cuando se hace entrega de los primeros análi-

sis de información, los algoritmos que están detrás del procesamiento continuo son considerados tediosos, por lo tanto, propensos al error, como se ha establecido anteriormente. Los Streaming Data, sin embargo, son mejor calificados en el tiempo de entrega de los análisis, en comparación a los procesos Batch.

Si entramos en el mundo de la estadística, el principal interés se encuentra, en el análisis de información de los modelos estadísticos. El ejemplo más importante, utilizado en esta tesis, son los modelos de Series de Tiempo ARIMA.

El modelo original ARIMA es procesado con toda la información disponible y realiza predicciones, es decir, que este Modelo ocupará el proceso Batch. El problema se presenta, cuando se entrega un nuevo dato al modelo y la cantidad de información es extensa, cayendo en errores y largos intervalos de tiempo. La única ventaja del proceso Batch en este caso, es que la estimación de parámetros es más precisa a diferencia del procesamiento continuo.

Debido a esta última problemática, nace el interés principal de esta investigación e implementación, Los Modelos ARIMA para Streaming Data. Como se mencionaba anteriormente, en las características de los procesos continuos, se debe dar inicio, con algún enfoque teórico para aproximarse al modelo original ARIMA (Anava O and Shamir, 2013) y algoritmos tediosos (Chenghao L and S, 2016), pero no presenta costo en términos de tiempo. Lo anterior, con el objetivo principal, de realizar nuevas iteraciones dentro de estos algoritmos por cada nuevo dato que entra, originando un nuevo proceso continuo que diferencia al proceso Batch, anteriormente mencionado, adquiriendo ventajas en términos de costos de tiempo y adaptándose a las nuevas demandas del mundo real.

3.3. Técnicas de Aprendizaje en Línea para Procesos ARIMA

Para el aprendizaje en línea o continuo con los Modelos ARIMA, seguiremos un típico marco de teoría de juegos, donde el jugador en línea secuencialmente, se compromete a una decisión y luego sufre de una pérdida desconocida, antes de tiempo. Puede ser contradictorio o incluso depender de las acciones realizadas por el individuo a cargo tomar decisiones (Cesa-Bianchi and G, 2006). En nuestro contexto, la configuración en línea para ARIMA, los vectores de coeficientes (α, β) son fijados por el adversario. En el momento t, el adversario elige el ruido ϵ_t y entonces genera la observación resultante X_t, basados en la ecuación 3.7 y ecuación 3.9.

Es importante tomar en cuenta, que los verdaderos valores de los vectores de coeficientes (α, β) y el ruido ϵ_t, no se relevan en cualquier momento t.

Considerando la iteración ARIMA en línea en el tiempo t, el jugador realiza una predicción $\tilde{X}_t$, y entonces el verdadero valor de X_t es relevado al aprendiz. Como resultado de esto, el jugador sufre una pérdida, esto denotado como $l_t(X_t, \tilde{X}_t)$.

Más formalmente, podemos definir la función de pérdida de la siguiente manera:

$$f_t(\alpha, \beta) = l_t(X_t, \tilde{X}_t(\alpha, \beta)) \tag{3.10}$$

$$f_t(\alpha, \beta) = l_t(X_t, \bigtriangledown^d \tilde{X}_t + \sum_{i=0}^{d-1} \bigtriangledown^i X_{i-1})$$

$$f_t(\alpha, \beta) = l_t(X_t, \sum_{i=1}^{p} \alpha_i \bigtriangledown^d X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \sum_{i=0}^{d-1} \bigtriangledown^i X_{i-1}).$$

Así pues $l_t(X_t, \tilde{X}_t(\alpha, \beta))$ representa la pérdida en el tiempo t, $\tilde{X}_t(\alpha, \beta)$ representa la predicción basados en los vectores de coeficientes (α, β) y $\bigtriangledown^d \tilde{X}_t$ representa la predicción del orden diferencial d de la observación en el tiempo t.

El objetivo del aprendizaje en línea ARIMA, es minimizar la suma de pérdidas durante un cierto número de rondas T. Más formalmente, podemos definir esta Pérdida del jugador después de T rondas como:

$$P_T = \sum_{i=1}^{T} l_t(X_t, \tilde{X}_t) - \min_{\alpha,\beta} \sum_{i=1}^{T} l_t(X_t, \tilde{X}_t(\alpha, \beta)) \tag{3.11}$$

Por lo tanto $l_t(X_t, \tilde{X}_t)$ representa la función de pérdida de la predicción real en el tiempo t y $l_t(X_t, \tilde{X}_t(\alpha, \beta))$ representa la función de pérdida del jugador en el tiempo t.

Nuestra meta es diseñar un algoritmo eficiente, que pueda garantizar que la Pérdida crece sub-linealmente en función de T, es decir, que a medida que aumente T en promedio la Pérdida se acerca a cero.

Dada esta función de pérdida, se define en la ecuación 3.10, se podría considerar aplicar algunas técnicas de optimización convexa en línea existentes, para estimar los vectores de coeficientes (α, β) para la tarea de aprendizaje en línea ARIMA. Sin embargo, esto no es posible, porque los términos de ruido ϵ_t son desconocidos en cualquier momento. Incluso, si se nos dan algunos valores de los coeficientes (α, β) no podríamos generar una predicción $\tilde{X}_t$. Está falta de información, hace que sea también difícil de calcular los mejores coeficientes en retrospectiva, de este modo competir contra el mejor modelo ARIMA, estaría mal.

3.3.1. Predicción Online ARIMA

Como se aclaró anteriormente, no podemos usar algoritmos de aprendizaje en línea convexas existentes en el espacio de vectores de los coeficientes (α, β), ya que los términos de ruido son desconocidos para nosotros en cualquier etapa. Para hacer frente a esto, se propone la siguiente idea de *Enfoque de Aprendizaje Incorrecto* (Anava O and Shamir, 2013), con el objetivo de diseñar una solución. La predicción

no viene directamente desde el modelo ARIMA, que intenta imitar al modelo subyacente, sino a partir de un modelo ARIMA modificado (sin los términos de ruido explícitos) que se aproximan al modelo original.

Más específicamente, se ajusta un proceso ARIMA(p+m,d,0) para aproximar al modelo original ARIMA(p,d,q), donde en cada punto t, elegimos un vector de coeficientes (p+m)-dimensional $\gamma \in \mathbb{R}^{p+m}$ con $m \in \mathbb{N}$ arbitrario, generando efectivamente la aproximación de la predicción original:

$$\tilde{X}_t(\gamma^t) = \sum_{i=1}^{p+m} \gamma_i \bigtriangledown^d X_{t-i} + \sum_{i=0}^{d-1} \bigtriangledown^d x_{t-1} \tag{3.12}$$

De ello se desprende que nuestra pérdida en la iteración t es determinada mediante la siguiente función de pérdida:

$$l_t^m = l_t(X_t, \tilde{X}_t(\gamma^t))$$

$$l_t^m = l_t(X_t, \sum_{i=1}^{p+m} \gamma_i \bigtriangledown^d X_{t-i} + \sum_{i=0}^{d-1} \bigtriangledown^i X_{t-1}) \tag{3.13}$$

En resumen, lo principal es elegir correctamente un valor adecuado para el parámetro m, y lo que será la pérdida con tales aproximaciones. Para este caso, se presenta un grupo de algoritmos online populares para el aprendizaje en línea, con el objetivo de optimizar solucionadores de ARIMA (Chenghao L and S, 2016): 1) Los algoritmos cuyo objetivo es explotar la información para acelerar la convergencia de la optimización, en el siguiente caso, el algoritmo ONS de Hazan (Hazan E and S, 2007). En la optimización convexa en línea, los enfoques se basan principalmente, en el primer orden de optimización; es decir, optimización utilizando la derivada de primer orden de la función de pérdida.

3.3.2. Definición y cálculo de los parámetros del Algoritmo

Antes de presentar los algoritmos, se deben introducir un par de anotaciones para definir los siguientes parámetros. Denotamos como $\mathcal{K}$ al conjunto de candidatos del vector de coeficientes (p+m)-dimensional que podemos elegir en cada iteración; se define como:

$$\mathcal{K} = \left\{ \gamma \in \mathbb{R}^{p+m}, |\gamma_j| \leq 1, j = 1, ..., m \right\} \tag{3.14}$$

También se denota el diámetro de $\mathcal{K}$ como $D = 2 * \sqrt{(p+m)}$.

A continuación, se denota por G como el límite superior de $|| \bigtriangledown l_t^m(\gamma)||$ para todo tiempo t y $\gamma \in \mathcal{K}$. Este parámetro depende de la función de pérdida considerando, de esta manera su cálculo. Lo que equivale a $G = 2 * \sqrt{(p+m)}(X_{max}^2)$ por la pérdida cuadrada.

Finalmente, denotamos por λ el parámetro exp-concavidad de las funciones de pérdida $\{l_t^m\}_{t=1}^T$, es decir, se cumple que $e^{-\lambda l_t^m(\gamma)}$ es cóncavo para todo t. Este parámetro es relevante para las funciones de pérdida exp-concavas y su cálculo se realiza también de acuerdo a la función de pérdida considerada. Para este caso específico, la pérdida cuadrada, se denota como $\lambda = \frac{1}{p+m}$.

3.3.3. Supuestos para algoritmos en procesos ARIMA

- Los términos de ruidos se generan de forma independiente, cada uno con distribución de media cero.
- La función l_t es L-Lipshitz continua con $L > 0$.
- Los coeficientes α_i satisfacen $|\alpha_i| < c$, con c $\in \mathbb{R}$.
- La señal esta delimitada i.e. $|x_t| < X_{max} \in \mathbb{R}, \forall t$.

3.4. Método Newton Step

El Método de Newton es aproximar la función f() a x_k por la cuadrático, definido por la segunda orden de la aproximación de Taylor (con desarrollo inexacto para t=0), dada por:

$$m_k : p \to f(x_k) + \bigtriangledown f_k^T p + p^T \frac{\partial^2 f}{\partial x^2}(x_k) p$$

Las dos funciones f() y $x \to m_k(x - x_k)$ tienen el mismo valor de su derivada de orden 0 al orden 2 en x_k.

Esta función tiene un mínimo global, que se obtiene cuando la derivada de m_k desaparece y que la dirección del descenso del método de Newton da:

$$p_k = -\frac{\partial^2 f}{\partial x^2}(x_k)^{-1} \bigtriangledown f_k$$

El método de Newton, tiene una longitud natural al paso $\alpha = 1$. Proporciona una convergencia cuadrática al mínimo local, bajo la condición que $\frac{\partial^2 f}{\partial x^2}$ es definida positiva, en cualquier punto de la secuencia de descenso. Esto se formaliza por el teorema del radio de convergencia de Newton.

3.4.0.1. Radio de convergencia de Newton

Se considera una función f() y una sección $\mathcal{N}$ de mínimo local x^* de f() continua L-Lipschitz. Entonces para cualquier punto x_0 es suficientemente cerca de x^*,$(x_k)_{k \in \mathcal{N}}$ converge cuadráticamente a x, dando origen a $0 < \rho < 1$ y $N \in \mathbb{N}$, suficientemente para que:

$$\forall k > N, ||f(x_{k+1})|| < \rho ||f(x_k)||^2$$

Por consiguiente, la taza de convergencia del radio ρ depende de la norma Hessiana y el coeficiente L-Lipschitz:

$$\rho = \frac{L}{||\frac{\partial^2 f}{\partial x^2}(x^*)||}$$

La continuidad de Lipschitz en torno a un punto, satisface la segunda condición de optimización, para la garantía definida positiva del Hessiano durante todo el descenso. Por el contrario, si el Hessiano se convierte en no positiva (i.e. x_0 está lejano de x^*), la dirección del descenso puede no tener una dirección de descenso más. En este caso, el algoritmo diverge, generalmente de forma violenta.

3.4.1. ARIMA Online Newton Step (ARIMA-ONS)

Principalmente, el algoritmo ONS, explota la segunda derivada de la función de pérdida. En otras palabras, el poder ver la optimización en línea análoga al Método de aprendizaje offline Newton-Rhapson (Ypma and J, 1995).

En este método, se muestra la propuesta de algoritmo ARIMA-ONS que iterativamente optimiza los vectores de coeficientes γ^t del modelo Online ARIMA, solucionado mediante Newton Step. Se debe tener en cuenta, que la notación $\Pi_{\mathcal{K}}^{A_t}$ se refiere a la proyección sobre $\mathcal{K}$ en la norma inducida por A_t, siendo A_t una matriz definida positiva y denotaremos $\|x\|_A = \sqrt{\langle x, Ax \rangle}$ como la norma Mahalanobis del vector x respecto a A, es decir:

$$\Pi_{\mathcal{K}}^{A_t}(y) = \arg\min_{x \in \mathcal{K}} (y - x)^T A_t (y - x) \triangleq \arg\min_{x \in \mathcal{K}} \|y - x\|_{A_t}. \tag{3.15}$$

Siendo A_t, una matriz (p+m)x(p+m) arbitraria, y que su inversa puede ser calculada eficientemente, después de cada actualización usando la formula Sherman-Morrison.

Por lo tanto, el algoritmo queda como:

Entrada;

Parámetros p, d, q; taza de aprendizaje η; Matriz inicial A_0 de (p+m)x(p+m).

para $t = 1$ **a** *T-1* **hacer**

Predecir $\tilde{X}_t(\gamma^t) = \sum_{i=1}^{p+m} \gamma_i \bigtriangledown^d X_{t-i} + \sum_{i=0}^{d-1} \bigtriangledown^i X_{t-1}$;

Recibir X_t y incurrir a la pérdida $l_t^m(\gamma^t)$;

Dejar $\bigtriangledown_t = \bigtriangledown l_t^m(\gamma^t)$;

Actualizar $A_t = A_{t-1} + \bigtriangledown_t \bigtriangledown_t^T$;

Conjunto $\gamma^{t+1} = \Pi_{\mathcal{K}}^{A_t}(\gamma^t - \frac{1}{\eta} A_t^{-1} \bigtriangledown_t)$

fin

Algoritmo 1: ARIMA-ONS(p,d,q)

Capítulo 4

Implementación de Métodos de Aprendizaje en línea

En este capítulo, se aborda los algoritmos de aprendizaje en línea, para los procesos ARIMA, en conjunto con el aprendizaje en lote, que en esta instancia se utilizará Yule-Walker, y además, se evaluará el rendimiento de cada uno de estos.

4.1. Diferenciación de procesos ARIMA

Como se menciona en Predicción Online ARIMA, se deben ajustar los procesos ARIMA(p+m,d,0) para la aproximación al modelo original ARIMA(p,d,q). Es decir, si $X_t \sim ARIMA(p,d,q)$ entonces, $\bigtriangledown^d X_t \sim ARMA(p,q)$, por lo tanto, X_t satisface:

$$\bigtriangledown^d X_t = \sum_{i=1}^{p} \alpha_i \bigtriangledown^d X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t, \tag{4.1}$$

Por lo tanto $\epsilon_t \overset{iid}{\sim} RB(0,\sigma^2)$, d es fijo y los coeficientes $\alpha_1, ..., \alpha_p$ y $\beta_1, ..., \beta_q$ son fijos.

De la misma manera, se puede reescribir este proceso, de una manera más compacta:

$$\Phi_p(B)(1-B)^d X_t = \Theta_q(B)\epsilon_t, \tag{4.2}$$

De modo que $\Phi()$ es el polinomio Autorregresivo, $\Theta()$ es el polinomio de Medias Móviles y $(1-B)^d$ representa las d veces que fue diferenciada la serie. Además, se debe tener en cuenta que las raíces del polinomio Autorregresivo, deben estar fuera del círculo unitario, para cumplir, que el proceso $\bigtriangledown^d X_t$ sea causal, de esta manera la condición de estacionalidad y a su vez, el polinomio $\Theta()$ debe ser invertible.

4.1.1. Procesos AR infinitos

Dado que $\bigtriangledown^d X_t$ es un proceso ARMA(p,q), lo denotaremos como Y_t y asumiremos que el polinomio $\Theta()$ es invertible, con cuyo único objetivo, es cumplir el proceso $Y_t \sim ARMA(p,q)$ que es equivalente al proceso $Y_t \sim AR(\infty)$, dónde:

$$\Pi_\infty(B) = \frac{\Phi_p(B)}{\Theta_q(B)} = (\Pi_0 + \Pi_1 + \Pi_2 + ...) \tag{4.3}$$

Por lo tanto, $Y_t \sim AR(\infty)$ queda como:

$$\sum_{j=0}^{+\infty} \Pi_j Y_{t-j} = \epsilon_t \tag{4.4}$$

Por lo tanto, debido a que tenemos el proceso $Y_t \sim AR(\infty)$, se puede definir el procesamiento Batch Yule-Walker, para estimar los parámetros y hacer posible la comparación.

4.2. Procesamiento en Batch

4.2.1. Método de Yule-Walker

Las ecuaciones de Yule-Walker, también conocidas como el método de los momentos, son aquellas que se igualan a los momentos teóricos y los momentos empíricos

i.e. El procedimiento consiste, en escribir los coeficientes de un proceso AR(p) en términos de las autocorrelaciones que son conocidas.

En este sentido formaremos las ecuaciones de Yule-Walker. sea $Y_t \sim AR(p)$ causal, es decir:

$$Y_t - \phi_1 Y_{t-1} - \phi_2 Y_{t-2} - ... - \phi_p Y_{t-p} = \epsilon_t \qquad \backslash Y_t$$
$$Y_t^2 - \phi_1 Y_{t-1} Y_t - \phi_2 Y_{t-2} Y_t - ... - \phi_p Y_{t-p} Y_t = \epsilon_t Y_t \qquad \backslash \mathbb{E}$$

Generando la primera ecuación,

$$\gamma(0) - \phi_1 \gamma(1) - \phi_2 \gamma(2) - ... - \phi_p \gamma(p) = \sigma_\epsilon^2 \tag{4.5}$$

Para las restantes ecuaciones hasta $y_{t-(p-1)}$ son,

$$Y_t - \phi_1 Y_{t-1} - \phi_2 Y_{t-2} - ... - \phi_p Y_{t-p} = \epsilon_t \qquad \backslash Y_{t-i}$$
$$Y_t Y_{t-i} - \phi_1 Y_{t-1} Y_{t-i} - \phi_2 Y_{t-2} Y_{t-i} - ... - \phi_p Y_{t-p} Y_{t-i} = \epsilon_t Y_{t-i} \qquad \backslash \mathbb{E}$$
$$\gamma(i) - \phi_1 \gamma(i-1) - \phi_2 \gamma(i-2) - ... - \phi_p \gamma(i-p) = 0$$

Generando (p+1) ecuaciones con la última,

$$\gamma(p) - \phi_1 \gamma(p-1) - \phi_2 \gamma(p-2) - ... - \phi_p \gamma(0) = 0$$

Entonces, se puede reescribir de forma matricial:

$$\begin{pmatrix} \gamma(1) \\ \gamma(2) \\ \vdots \\ \gamma(k) \end{pmatrix}_{1xp} - \begin{pmatrix} \gamma(0) & \gamma(1) & \cdots & \gamma(p-1) \\ \gamma(1) & \gamma(0) & \cdots & \gamma(p-2) \\ \vdots & \cdot & \cdot & \vdots \\ \gamma(p-1) & \cdot & \cdot & \gamma(0) \end{pmatrix}_{pxp} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{pmatrix}_{1xp} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{1xp}$$

En este sentido, sea $\Gamma = [\gamma_{(i-j)}]_{pxp}$, $\gamma = [\gamma_{(i)}]_{1xp}$ y $\phi = [\phi_{(i)}]_{1xp}$ con i,j=1,...,p, entonces se puede calcular:

$$\hat{\phi} = \Gamma^{-1} * \gamma \tag{4.6}$$

Siendo $\hat{\phi}$ el estimador de Yule-Walker de ϕ.

4.3. Procesamiento en Stream

4.3.1. Aspectos teóricos en la implementación para ARIMA-ONS

Puesto que, ya fue mencionado con anterioridad, los aspectos teóricos del algoritmo. En esta instancia, mostramos de manera clara los aspectos teóricos para implementar este algoritmo, utilizando los principales resultados teóricos de Anava (Anava O and Shamir, 2013) :

- La taza de aprendizaje es $\eta = \frac{1}{2}\,\text{mín}\{4GD, \lambda\}$.
- El $\epsilon = \frac{1}{\eta^2 D^2}$
- La matriz arbitraria es $A_0 = \epsilon I_{p+m}$

Siento estos, garantizar la satisfacción del algoritmo ARIMA-ONS que:

$$\sum_{t=1}^{T} l_t^m(\gamma^t) - min_{\alpha,\beta} \sum_{t=1}^{T} \mathbb{E}[f_t(\alpha, \beta)] = O((GD + \frac{1}{\lambda}) * log(T))$$

4.4. Comparación de Resultados en la implementación

Para probar el rendimiento de cada uno, se trabajará con series simuladas, provenientes de distintos procesos ARIMA, con el fin de tener control de los parámetros, y asi comprobar si las estimaciones, son cercanas al valor real del parámetro.

Para mostrar de manera más clara la comparación, se trabajará de la siguiente manera:

- Se realizarán las simulaciones de los procesos ARIMA, con datos distribuidos como $N(\mu, \sigma^2)$.
- El aprendizaje en línea, se iniciará desde la implementación de los algoritmos que se obtienen los RMSE y la Función de Pérdida en el tiempo T para cada simulación realizada.
- Para el aprendizaje en lote, se realizará la predicción de Yule-Walker en el tiempo T, mediante un ajuste entre el tiempo 1 al T-1.
- Estos últimos 2 pasos, se realizarán con diferentes tamaños de simulaciones y entre 20 a 100 repeticiones para la estabilidad.
- Se graficarán y/o se simplificarán en tablas los resultados para comparar la capacidad predictiva de ARMA-ONS y Yule-Walker.

4.4.1. RMSE

La raíz del cuadrático medio, es una de las medidas más utilizadas para comparar valores predichos de un modelo y los valores observados. El RMSE viene dado por:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(X_t - \hat{X}_t)^2}{T}}$$

Siendo $\hat{X}_t$ la predicción y X_t la señal en el tiempo t.

4.5. Implementación en Software

Para la aplicación de los algoritmos propuestos, las simulaciones de los datos y la estimación de Yule-Walker, se utilizan los paquetes de R.

4.5.1. Simulaciones

- **Arima.sim**: Simulación de un modelo arima.

```
arima.sim(model,n,rand.gen=rnorm,innov=rand.gen(n,...),
n.start=NA,start.innov=rand.gen(n.start,...),...)
```

Argumentos:

- Model: Una lista (p,d,q) con los coeficientes AR y MA.
- n: Largo de la serie.
- rand.gen: una función para generar las innovaciones.
- innov: una serie de innovaciones opcionales.
- n.start: duración del período de "burn-it".
- start.innov: una serie temporal opcional de innovaciones que se utilizarán para el período de "burn-it".

4.5.2. Métodos de aprendizaje en línea

- **ARIMA.ONS**: Algoritmo proveniente de Newton-Step para los procesos ARIMA. Esta función estima los parámetros y su Pérdida Cuadrática en cada tiempo, entregando finalmente los parámetros en el tiempo T y la función de Pérdida Cuadrática en todo t.

```
ARIMA.ONS(Serie,p,q,d).
```

Argumentos:

- Serie: Serie entregada para realizar las estimaciones y obtener la Pérdida Cuadrática.
- p: Orden de AR(p).
- q: Orden de MA(q).
- d: Cantidad de veces que se ha diferenciado.

- **ARMA.ONS**: Algoritmo proveniente de Newton-Step para los procesos ARMA. Esta función estima los parámetros y su Pérdida Cuadrática en cada tiempo, entregando finalmente los parámetros en el tiempo T y la función de Pérdida Cuadrática en todo t.

```
ARIMA.ONS(Serie,p,q).
```

Argumentos:

- Serie: Serie entregada para realizar las estimaciones y obtener la Pérdida Cuadrática.
- p: Orden de AR(p).
- q: Orden de MA(q).

4.5.3. Método Yule-Walker

- **ar**: Ajuste para procesos AR.

```
ar(x, aic = TRUE, order.max = NULL,
   method = c(``yule-walker", ``burg", ``ols", ``mle", ``yw"),
   na.action, series, ...)
```

Argumentos:

- x: Una serie univaridada o multivariada.
- aic: Si es TRUE, entonces el orden se determinará mediante el AIC. Si es FALSE, el modelo ajustará el orden dado el argumento order.max.
- order.max: Order máximo a ajustar.
- method: Especifica el método a ajustar el modelo.
- na.action: Función que se llamará para manejar valores perdidos.
- Series: Nombres de las series.

Capítulo 5

Resultados

En este capítulo, se exponen los resultados obtenidos mediante simulaciones, comparando el método batch Yule-Walker, frente al algoritmo Newton-Step.

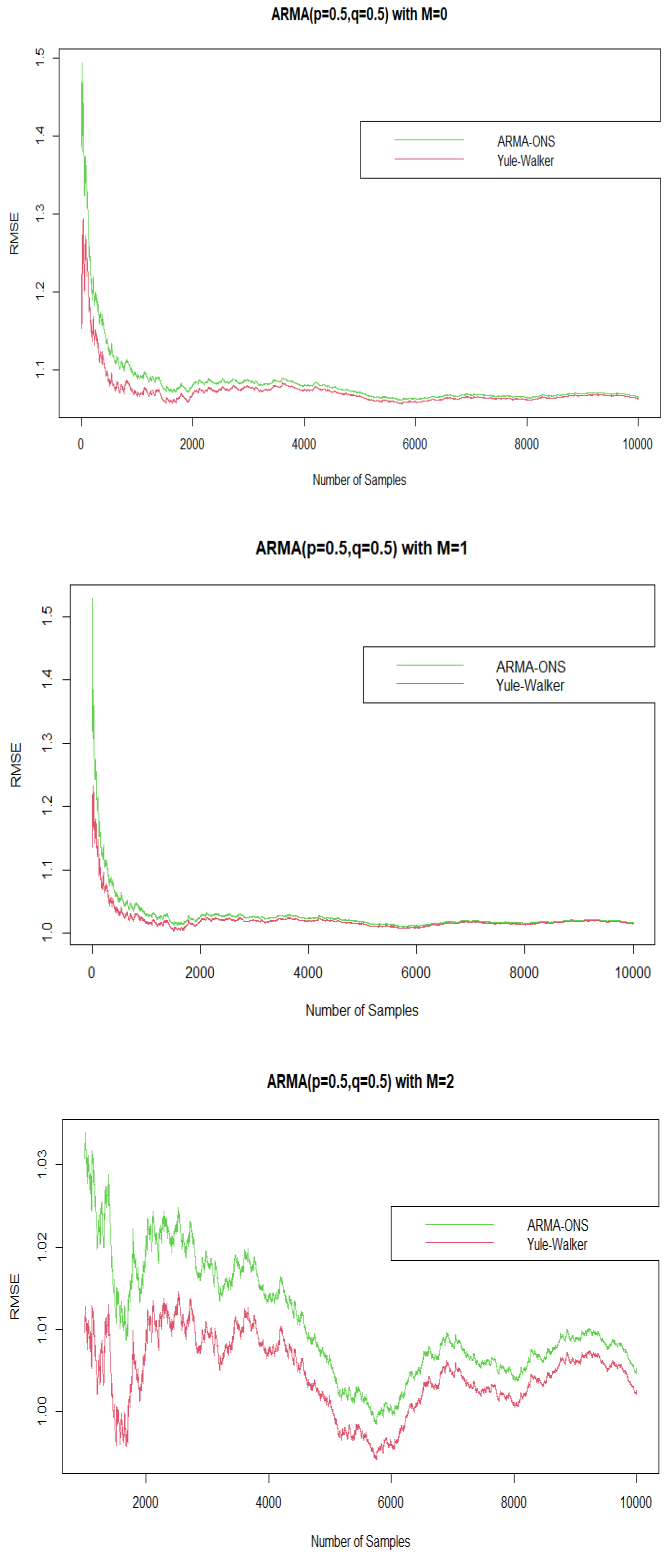
5.1. Series Estacionarias

5.1.1. Diferentes valores de M con coeficientes fijos

Se comienza generando un proceso estacionario ARMA, con coeficientes $\alpha = (0{,}8)$ y $\beta = (0{,}5)$, y términos de ruido no correlacionados con una distribución $N(0, 0{,}9^2)$.

En este caso, se muestran los gráficos resultantes de los RMSE con un tamaño de 10.000 en la simulación y el promedio de 20 repeticiones para la estabilidad.

Figura 5.1: RMSE con M=0, 1 y 2 para ARMA



Se puede destacar, que a medida que aumente el orden de estimación, los RMSE disminuyen levemente. Además, se distingue que el error de predición del método Yule-Walker es levemente menor que ARMA.ONS, para esta serie con parámetros fijos a través del tiempo.

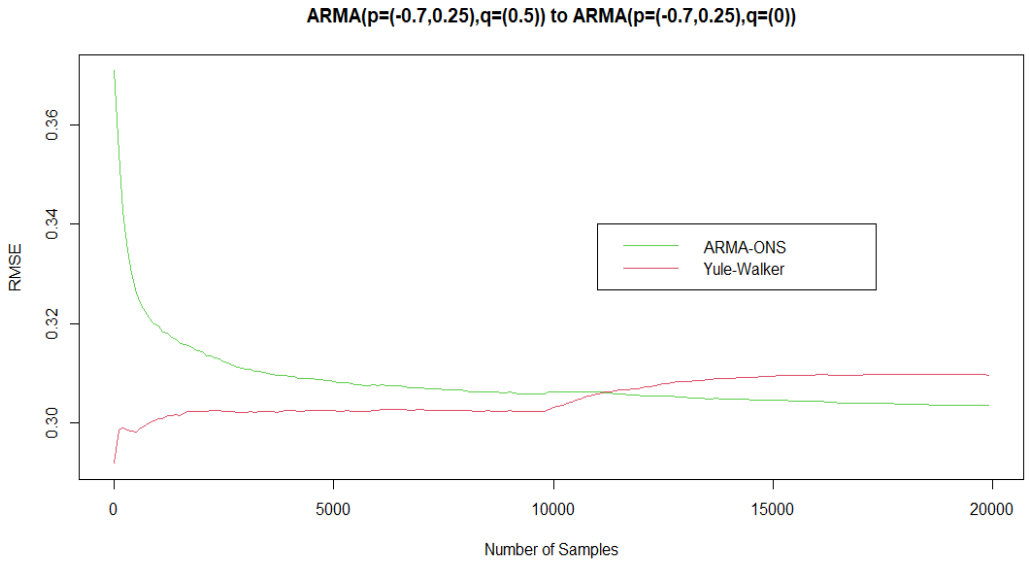
5.1.2. Cambio de coeficientes a través del tiempo

5.1.2.1. De ARMA a AR

En esta sección, se realiza simulaciones de un proceso estacionario ARMA de tal manera, en el primer set de coeficentes es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0{,}5)$ y el segundo set es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestra el gráfico resultante de los RMSE para un tamaño de 20.000 y el promedio de 20 repeticiones para la estabilidad.

Figura 5.2: Gráfico RMSE ARMA a AR



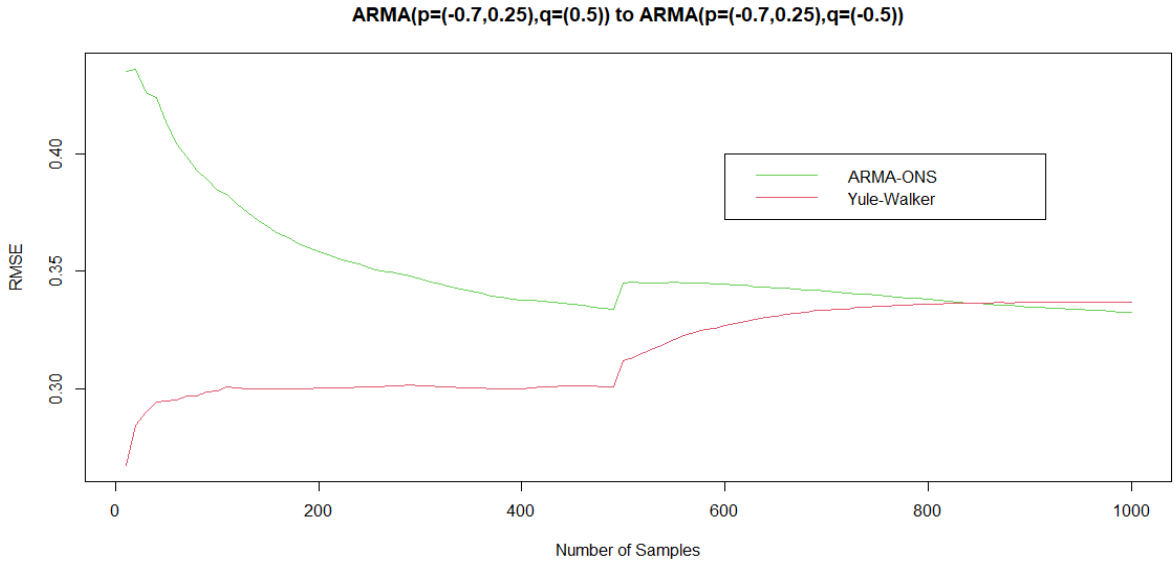
Con el fin de analizar el gráfico 5.2, se destaca, al realizar el cambio de coeficientes en t=10.000, la curva RMSE de Yule-Walker empieza a aumentar, a diferencia del ARMA.ONS disminuye.

5.1.3. Cambios de signo del coeficiente MA

En esta sección, se realiza simulaciones de un proceso estacionario ARMA de tal manera, en el primer set de coeficentes es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0{,}5)$ y el segundo set es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (-0{,}5)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestra el gráfico resultante de los parámetros con un tamaño de 1.000 en la simulación y el promedio de 100 repeticiones para la estabilidad.

Figura 5.3: Gráfico RMSE cambio de signo ARMA



Con el propósito de analizar el gráfico 5.3, se puede destacar, cuando se realiza el

cambio de coeficientes en t=500, la curva RMSE de Yule-Walker empieza a aumentar, que a diferencia del ARMA.ONS aumenta pero después disminuye.

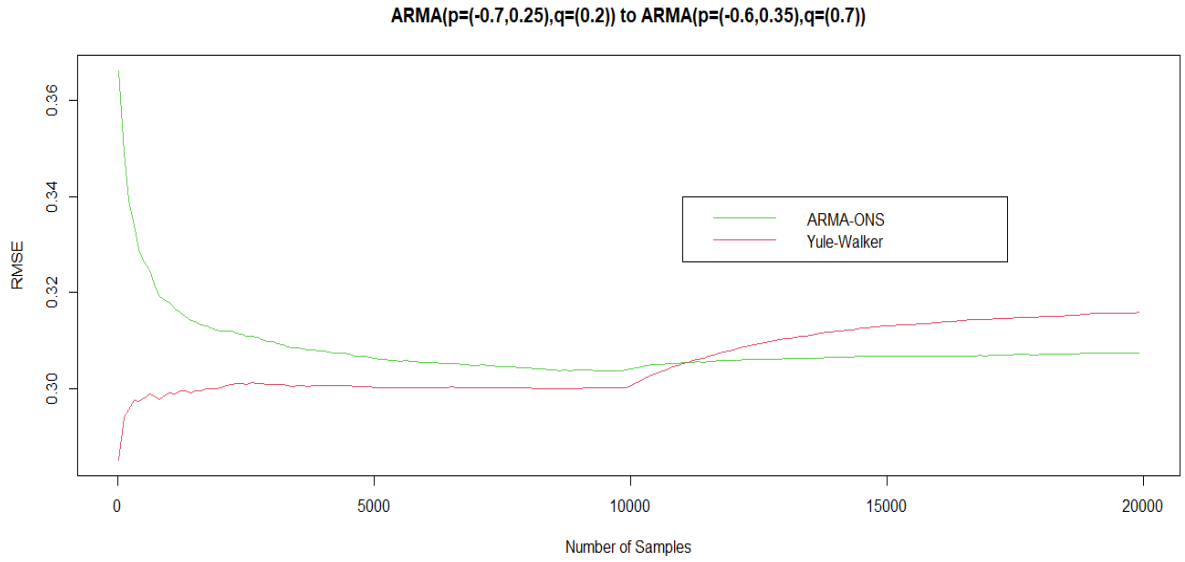
5.1.4. Leves cambios en los coeficientes

En esta sección, se realiza simulaciones de un proceso estacionario ARMA de tal manera, en el primer set de coeficentes es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0{,}2)$ y el segundo set es $\alpha = (-0{,}6, 0{,}35)$ y $\beta = (0{,}7)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestran el gráfico resultante de los RMSE, con un tamaño de 20.000 en la simulación y el promedio de 20 repeticiones, para la estabilidad.

Además, podemos respaldar que ONS es mejor ante este cambio mediante el gráfico de RMSE.

Figura 5.4: Gráfico RMSE leves cambios ARMA



Finalmente, analizando el gráfico 5.4, se puede destacar que cuando se realiza el cambio de coeficientes en t=10.000, la curva RMSE de Yule-Walker empieza a aumentar, que a diferencia del ARMA.ONS se mantiene.

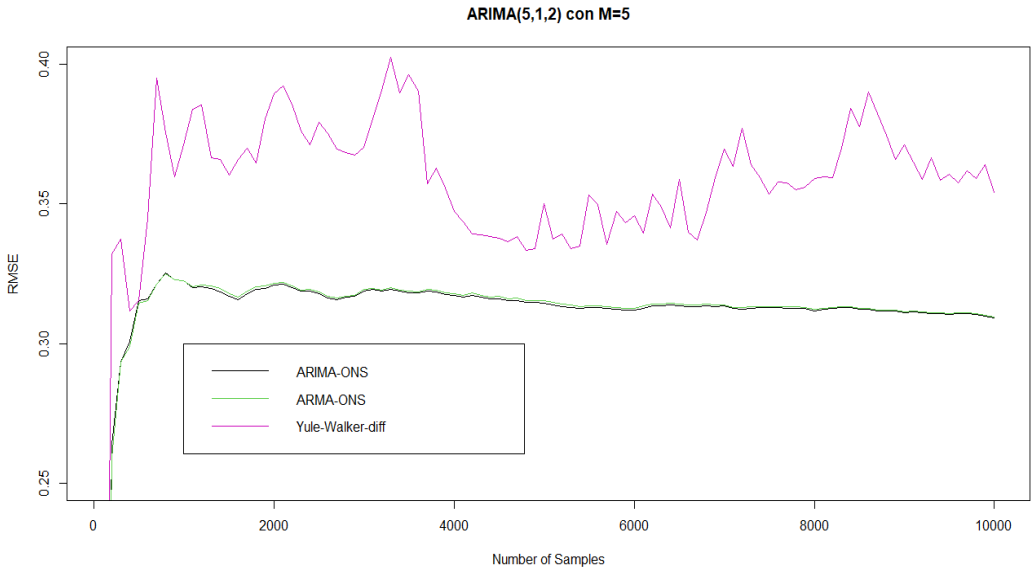
5.2. Series No Estacionarias

5.2.1. Parámetros fijos a través del tiempo

Se comienza generando un proceso no estacionario ARIMA(5,1,2), con coeficientes $\alpha = (0{,}6, -0{,}5, 0{,}4, -0{,}4, 0{,}3)$ y $\beta = (0{,}3, -0{,}2)$, y términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

Para el primer caso, se muestra el gráfico resultante de los RMSE con un tamaño de 10.000 en la simulación y el promedio de 5 repeticiones para la estabilidad.

Figura 5.5: Parámetros fijos tamaño 10.000

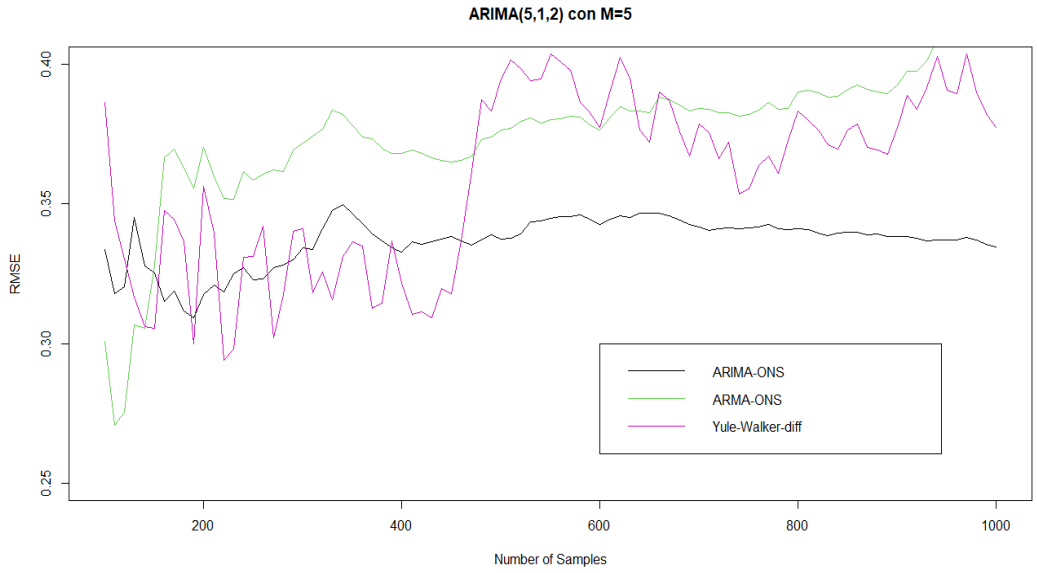


Se distingue que el error de predición del método Yule-Walker es mayor que

ARIMA.ONS y ARMA.ONS para toda esta serie con parámetros fijos a través del tiempo. Además, el RMSE de ARMA.ONS es levemente mayor que ARIMA.ONS.

Para el segundo caso, se muestra el gráfico resultante de los RMSE con un tamaño de 1.000 en la simulación y el promedio de 5 repeticiones para la estabilidad.

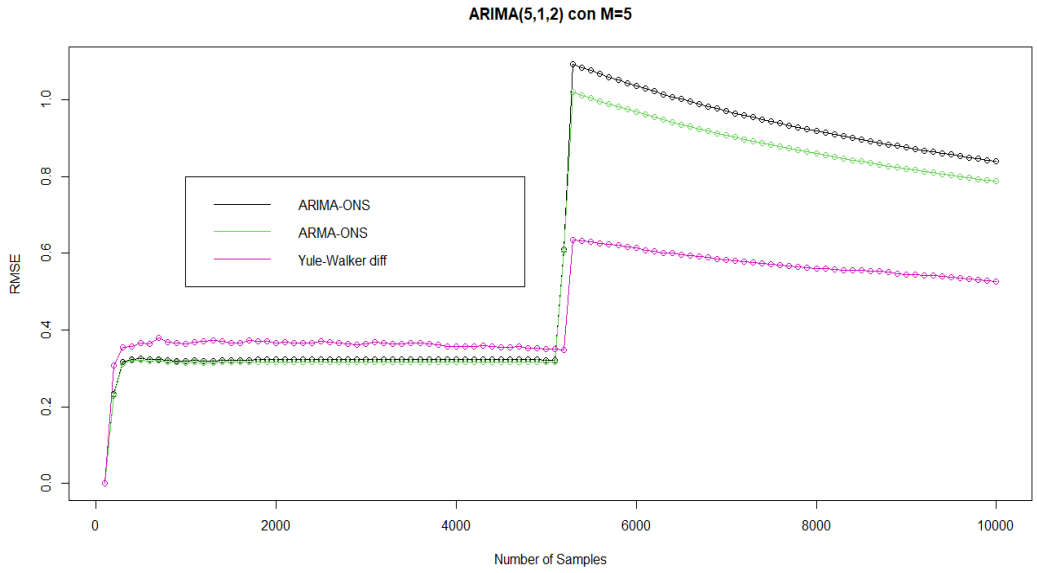
Figura 5.6: Parámetros fijos tamaño 1.000



Se distingue que el error de predición del método Yule-Walker es mayor que ARIMA.ONS para cuando la serie es mayor a los 450 datos aproximadamente con parámetros fijos a través del tiempo. Además, el RMSE de ARMA.ONS es mayor que ARIMA.ONS en toda la serie.

Para el tercer caso, se muestra el gráfico resultante de los RMSE con un tamaño de 10.000 en la simulación y el promedio de 20 repeticiones para la estabilidad, que a diferencia de los otros se tiene un dato atípico en el tiempo t=5.000.

Figura 5.7: Parámetros fijos tamaño 10.000



Se distingue que el error de predición del método Yule-Walker es mayor, para cuando la serie es menor a los 5.000 datos. Pero, cuando se ingresa un dato atípico, la estimación de los métodos de aprendizaje, son sensibles ante este cambio, provocando un error de predicción mayor que Yule-Walker.

5.2.1.1. Diferentes valores de M

Se comienza generando un proceso no estacionario ARIMA(5,1,2), con coeficientes $\alpha = (0{,}6, -0{,}5, 0{,}4, -0{,}4, 0{,}3)$ y $\beta = (0{,}3, -0{,}2)$, y términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestran los gráficos resultantes de los RMSE con un tamaño de 10.000 en la simulación y el promedio de 5 repeticiones para la estabilidad.

Figura 5.8: RMSE con M=0, 2 para ARIMA y ARMA

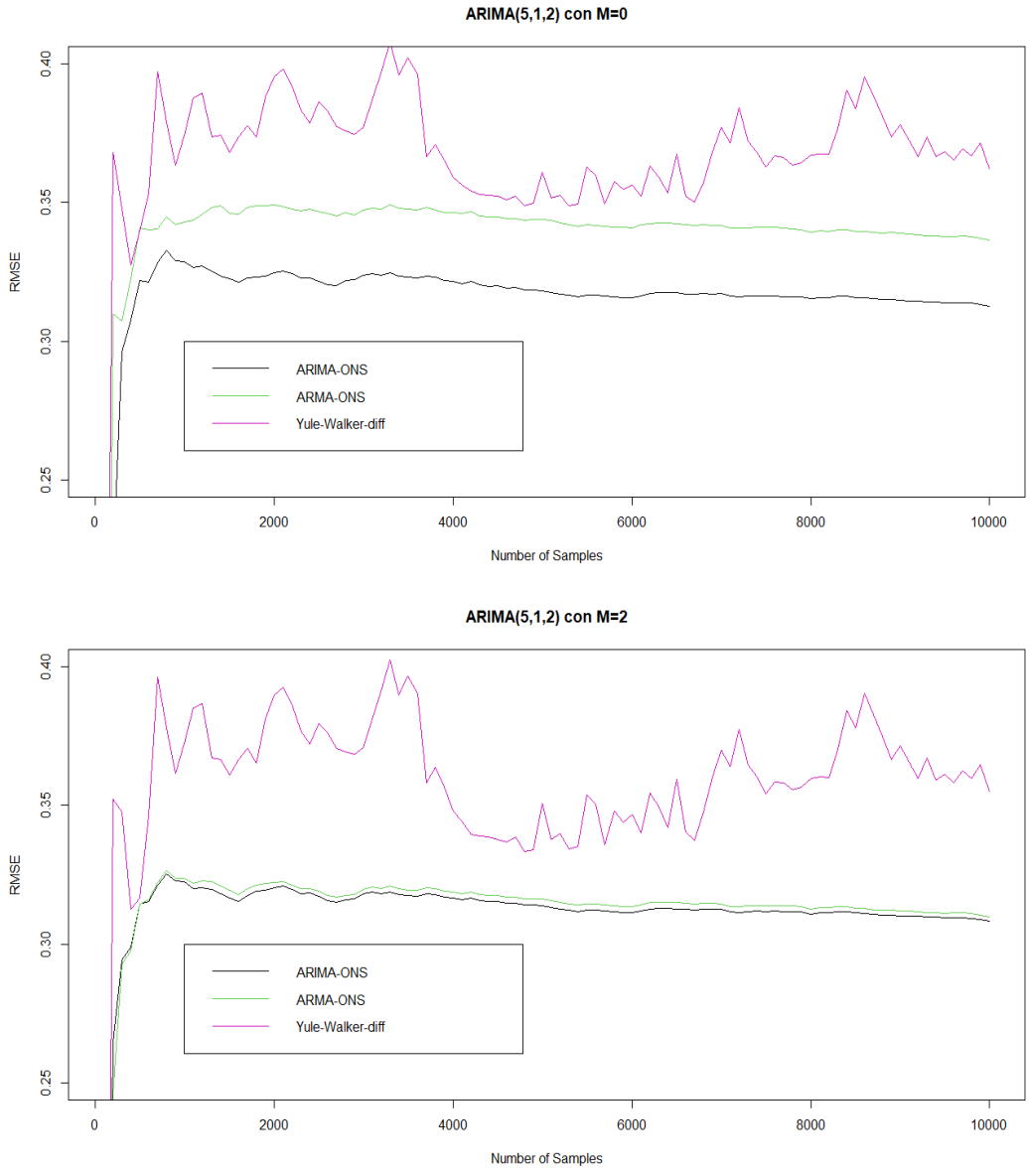
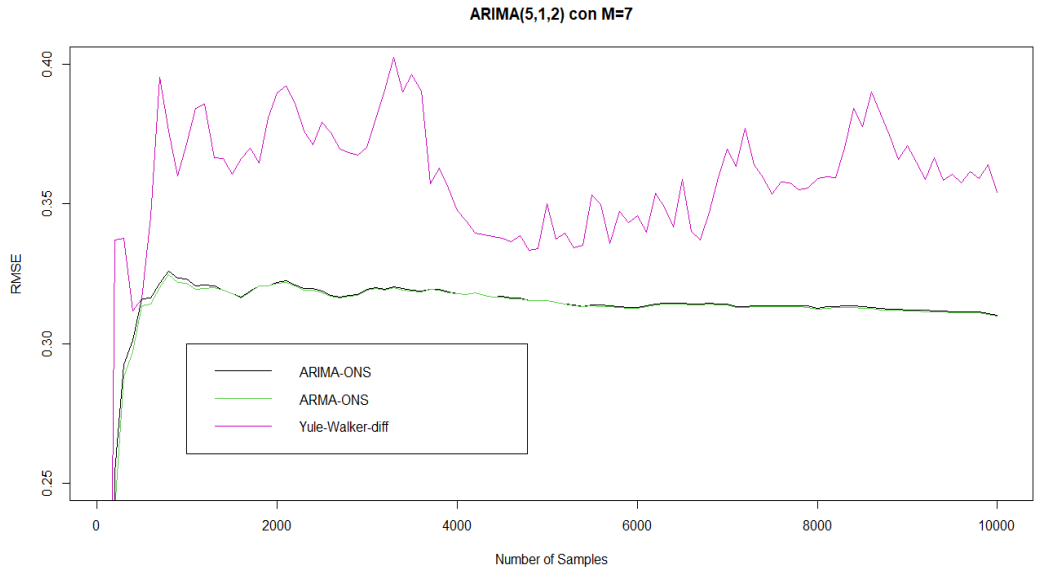


Figura 5.9: RMSE con M=7 para ARIMA y ARMA



Se puede destacar, que a medida que aumente el orden de estimación, los RMSE disminuyen para ARMA.ONS. Además, se distingue que el error de predicción del método Yule-Walker, es mayor para toda esta serie con parámetros fijos a través del tiempo.

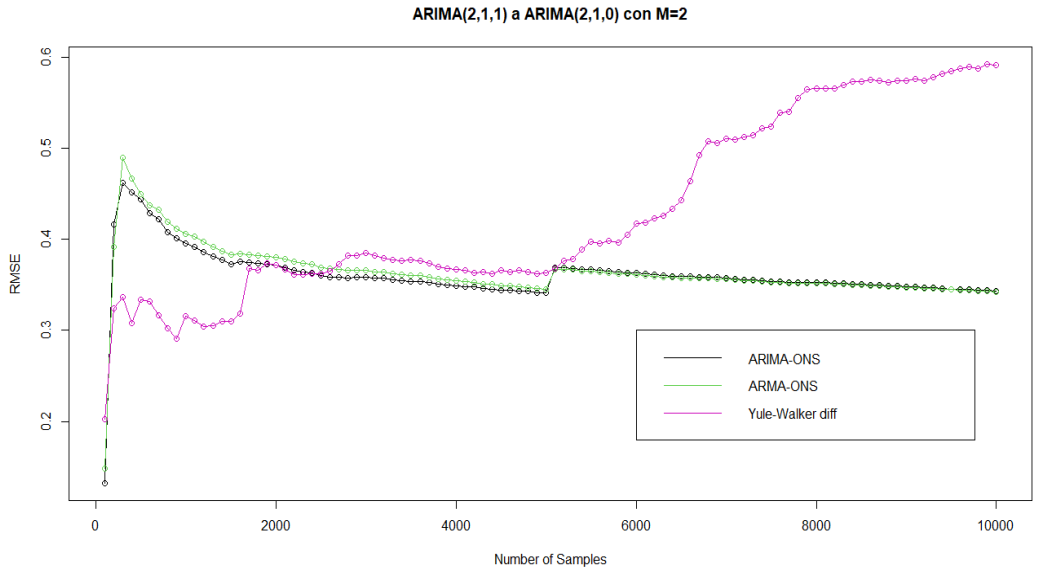
5.2.2. Cambio de coeficientes a través del tiempo

5.2.2.1. De ARIMA(2,1,1) a ARIMA(2,1,0)

En esta sección, se realiza simulaciones de un proceso no estacionario ARIMA(2,1,1) con M=2 de tal manera, en el primer set de coeficientes es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0{,}5)$ y el segundo set es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestra el gráfico resultante de los RMSE para un tamaño de 10.000 y el promedio de 20 repeticiones para la estabilidad.

Figura 5.10: Gráfico RMSE ARIMA(2,1,1) a ARIMA(2,1,0)



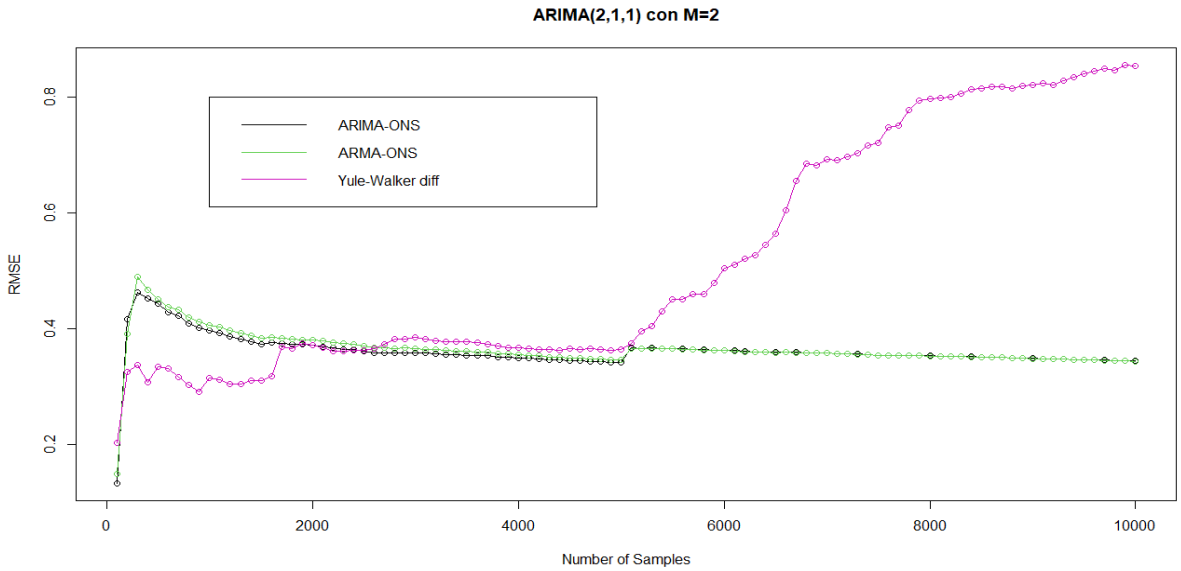
Se puede distinguir, que la curva del RMSE de los métodos de aprendizajes van disminuyendo a través del tiempo, que a diferencia de Yule-Walker, este va aumentando. Además, ante la eliminación del parámetro en las medias móviles, el error de predicción de Yule-Walker, va aumentando a través del tiempo.

5.2.2.2. Cambios de signo del coeficiente MA

En esta sección, se realiza simulaciones de un proceso no estacionario ARIMA(2,1,1) con M=2 de tal manera, en el primer set de coeficentes es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (0{,}5)$ y el segundo set es $\alpha = (-0{,}7, 0{,}25)$ y $\beta = (-0{,}5)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestra el gráfico resultante de los parámetros con un tamaño de 10.000 en la simulación y el promedio de 20 repeticiones para la estabilidad.

Figura 5.11: Gráfico RMSE cambio de signo ARIMA



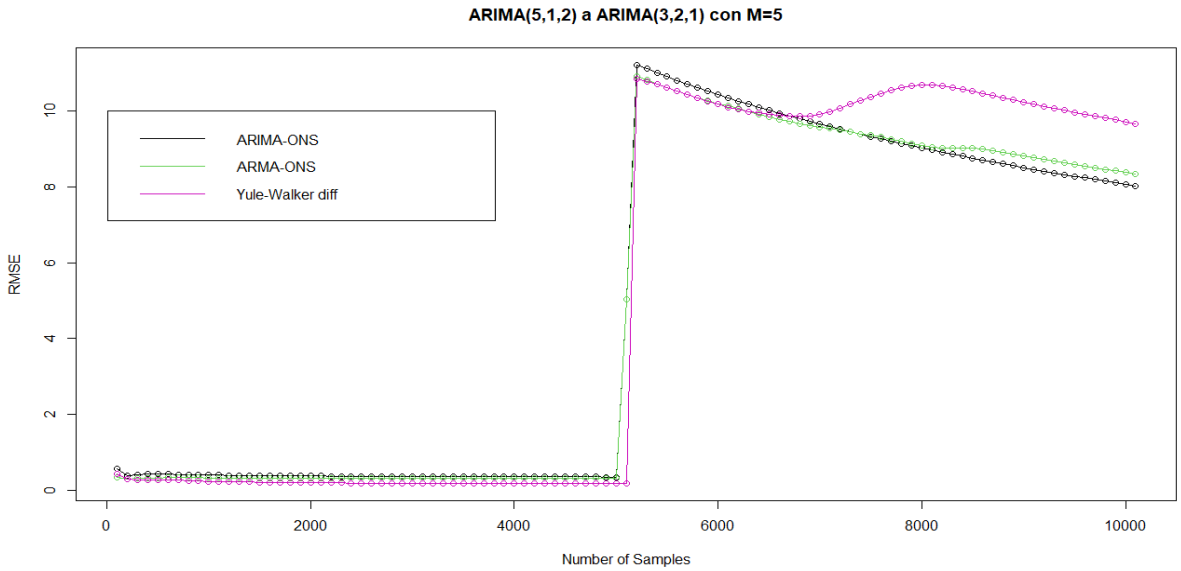
Con el proposito de analizar el gráfico 5.11, se puede destacar, cuando se realiza el cambio de signo en el coeficiente de medias móviles en t=5.000, la curva RMSE de Yule-Walker empieza a aumentar, que a diferencia del ARMA.ONS se mantiene constante.

5.2.2.3. Abrupto cambio de d

En esta sección, se realiza simulaciones de un proceso no estacionario ARIMA(5,1,2) a ARIMA(5,2,2) de tal manera, en el primer set de coeficentes es $\alpha = (0{,}6, -0{,}5, 0{,}4, -0{,}4, 0{,}3)$ y $\beta = (0{,}3, -0{,}2)$ y el segundo set es $\alpha = (0{,}5, -0{,}3, 0{,}1)$ y $\beta = (-0{,}2)$ y con términos de ruido no correlacionados con una distribución $N(0, 0{,}3^2)$.

En este caso, se muestran el gráfico resultante de los RMSE, con un tamaño de 10.000 en la simulación y el promedio de 20 repeticiones, para la estabilidad.

Figura 5.12: Gráfico RMSE con abrupto cambio de d



Finalmente, analizando el gráfico 5.12, se puede destacar que cuando se realiza el cambio de coeficientes en t=5.000, las curvas RMSE de los métodos suben considerablemente. Se distingue, que después del cambio de d, el RMSE de ARIMA.ONS y ARMA.ONS va disminuyendo más rápido comparado a Yule-Walker.

Capítulo 6

Conclusiones

En primer lugar, este trabajo se desarrolló para el enfoque, en distintas Series de Tiempo con aprendizaje en línea. Es importante destacar, que las predicciones de estas series son cercanas al método Batch, independiente de su función de pérdida.

Por otra parte, se considera importante señalar que los parámetros estimados con el algoritmo ARIMA.ONS en streaming data, convergen a los parámetros del método en lotes Yule-Walker, tanto para parámetros autorregresivos p iniciales y los M agregados.

Además, vale destacar que cuando los coeficientes tienen cambios a través del tiempo y estos afectan de manera negativa la convergencia de los parámetros en el método Yule-Walker, provoca un mayor error de predicción comparado al método ARIMA.ONS. Se puede agregar, que mientras mayor es el tamaño muestral, el error de predicción va aumentando para el método Batch en comparación al método ARIMA.ONS.

También, se puede mencionar que el método de aprendizaje en línea es sensible ante los datos atípicos, generando mayores errores de predicción en comparación al método en lote.

En el caso de ARMA.ONS, presenta distintos rendimientos con respecto de ARI-

MA.ONS en términos de convergencia, esto se debe principalmente a las proyecciones euclidianas implementadas en el algoritmo. En este sentido, si se mejora las proyecciones, los métodos convergen más rápido a los parámetros, reflejados en su error de predicción.

Por último, se puede señalar que los métodos de aprendizaje en línea, son llamativos, para los cambios bruscos o leves de los datos a través del tiempo, siendo fácil de ver en la curva de error de predicción.

Capítulo 7

Anexos

7.1. Función ARIMA.ONS

```
ARIMA.ONS<-function(Serie,p,d,q,m,pred)
{ nt<-length(Serie)-pred #Numero de rondas que se desea
  k<-p #Modelo AR
  Xmax<-max(Serie)
  minimo<-p+m+1
  D<-2*sqrt((m+k)) #Diametro de k <- dimension de gamma
  G<-2*sqrt(m+k)*(Xmax)^2 #Limite superior de gamma
  Lt<-matrix(nrow = (nt),ncol = (minimo-1))
  L<-as.matrix(runif(minimo-1, min=-0.5, max=0.5))
  Rate<-0.5*min(1/(m+k),4*D*G) #Radio
  Epsilon<-1/((Rate^2)*(D^2))
  A<-diag(Epsilon, m+k, m+k) #Matriz arbitraria A
  T<-nt #Rondas
  Tp<-pred
  x.p<-rep(0,T) #Ajuste, only ceros T veces
  loss<-rep(0,T) #Error del ajuste
  errp<-rep(0,Tp) #Error de prediccion
  #######      Train Data      ######
```

```
#Funcion Train
for(t in (d+1):T)
{x.t<-0 #Acumulacion del valor estimado
  for(i in 1:(m+k))
  {if(t-i-d<1)
    { break
    }else{
      if(d==0)
      {x.t<-x.t+L[i]*(Serie[t-i]) #Prediccion
      }else if(d==1)
      {x.t<- x.t+L[i]*(Serie[t-i]-Serie[t-i-1]) #Prediccion
      }else if(d==2)
      {x.t<- x.t+L[i]*(Serie[t-i]-2*Serie[t-i-1]+Serie[t-i-2])
      }else{
        x.t<- x.t+L[i]*(Serie[t-i]-3*Serie[t-i-1]+3*Serie[t-i-2]-Serie
            [t-i-3])
      }} }
  if(t-d<1)
  { break
  }else{ if(d==1)
  { x.t<- x.t+Serie[t-1]
  } else if(d==2)
  { x.t<- x.t+2*Serie[t-1]-Serie[t-2]
  }else{
    x.t<- x.t+3*Serie[t-1]-3*Serie[t-2]+Serie[t-3]
  }}
  x.p[t]<-x.t #Valor estimado
  loss[t]=(Serie[t]-x.t)^2 #Perdida cuadratica
  nabla=matrix(0, k+m, 1)
  for (i in 1:(k+m))
  { if(t-i-d>=1)
    {if(d==0)
      { x<-(Serie[t-i]) #Prediccion
```

```
51          }else  if(d==1)
52          {  x<-(Serie[t-i]-Serie[t-i-1]) #Prediccion
53          }else  if(d==2)
54          {x<-(Serie[t-i]-2*Serie[t-i-1]+Serie[t-i-2])
55          }else{x<-(Serie[t-i]-3*Serie[t-i-1]+3*Serie[t-i-2]-Serie[t-i-3])
                 } nabla[i,1]=-2*(Serie[t]-x.t)*x #Gradiente
56        }else{x=0
57          nabla[i,1]=-2*(Serie[t]-x.t)*x
58        }  }
59      A<- A+nabla*t(nabla)
60      L<- L-(1/Rate)*solve(A)*nabla
61      for(j  in  1:(minimo-1))
62      {  Lt[t,j]<- L[j,1]  }}
63    ########       Test Data        ######
64    offset.original=length(x.p)+1  #Salida del train
65    x.pp<-as.vector(rbind(x.p,rep(0,Tp)))
66    for(t  in  offset.original:(offset.original+Tp-1))
67    {x.t<-0
68      for(i  in  1:(m+k))
69      {x.t=x.t+L[i]*Serie[t-i]
70      }  x.pp[t]=x.t
71      errp[t-offset.original+1]<-(Serie[t]-x.t)^2}
72    RMSEA<-sqrt(mean(loss))  #Error de ajuste
73    RMSEP<-sqrt(mean(errp))  #Error de prediccion
74    #Resultados
75    var<-list(loss=loss,RMSEA=RMSEA,RMSEP=RMSEP,
76              Lt=Lt,x.p=x.p)
77    return(var) }
```

Bibliografía

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. *In Selected Papers of Hirotugu Akaike*, pages 199–213.

Anava O, Hazan E, M. S. and Shamir, O. (2013). Online learning for time series prediction. *In Conference on Learning Theory*, pages 172–184.

Brockwell, P. and R, D. (2009). *Time series: theory and methods*. Springer Science & Business Media.

Bubeck, S. (2011). Introduction to online optimization. *Princeton University*.

Cesa-Bianchi, N. and G, L. (2006). Prediction, learning and games. *Cambridge University Press*.

Chenghao L, Steven H, P. Z. and S, J. (2016). Online arima algorithms for time series prediction. *Thirtieth AAAI Conference on Artificial Intelligence*, pages 1867–1873.

Elorrieta, F. (2018). *Classification and Modeling of time series of astronomical data*. Pontificia Universidad Católica de Chile.

Gao J, Sultan H, H. J. and W, T. (2010). Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: a comparison. 17(3):237–240.

Hamilton, J. (1994). Time series analysis. 2.

Hazan E, A. A. and S, K. (2007). Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2–3):169–192.

Hoi S C, W. J. and P, Z. (2014). Libol: A library for online learning algorithms. *The Journal of Machine Learning Research*, 15(1):495–499.

Rabiner, L. and R, S. (2011). Digital speech processing. 6:237–258.

Schwarz, G. (1978). Estimating the dimension of a model. 6(2):461–464.

Stoffer, D. (2020). *Applied Statistical Time Series Analysis*.

Team, R. C. (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ypma and J, T. (1995). Historical development of the newton-raphson method. 37(4):531–551.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *International Conference on Machine Learning*, pages 928–935.