

Presentación 6

Series de Tiempo

Minería de Datos

José Alfredo Méndez

Luis Manuel Román García

Marcos Chávez

María Fernanda Mora Alba

Stephane Keil Rios

Maestría en Ciencias en Computación

Maestría en Ciencia de Datos

Profr. Dr. Ángel Kuri

Noviembre, 2015

Contenido

Introducción

Proyecciones en el tiempo (estructura de las BDs)

Modelos de variables

Modelos de redes neuronales para la predicción de series temporales

Modelos polinomiales no lineales

Comparación de ambos modelos

Conclusiones

Referencias

Introducción

“...but time and chance happeneth to them all.”

- Ecclesiastes

La importancia del estudio de las series de tiempo ha sido enfatizado a lo largo de los años por distintos autores Hooker[5], Einstein[4], Wiener[6]. Esto se debe a que este tipo de relaciones temporales surgen en una gran variedad de contextos, tanto científicos como sociales, por ejemplo: en el campo de la física, las finanzas, la biología, demografía, sociología, etc. No resulta una sorpresa entonces que en uno de sus trabajos, Tufte, reconociera a la visualización de series de tiempo cómo el tipo de gráfica más frecuente. En este mismo sentido y dada la variedad de fuentes de investigación como campos de aplicación que tratan con estas, el campo se ha llenado de un argot sofisticado con términos como tendencias, estacionalidad, periodicidad, autocorrelación, ciclicidad, efectos estacionales, control, asociación, etc. Por estas razones, el objetivo de esta sección es definir lo que es una serie de tiempo en el contexto más general posible, así mismo, se tratará de exemplificar la importancia de estas; tanto del punto de vista teórico, en el campo de los procesos estocásticos, como del punto de vista práctico, en el campo de la estadística aplicada. De la misma manera, se buscará identificar las principales propiedades que una serie de tiempo puede poseer y se explicara cómo es que estas son utilizadas para modelar su comportamiento. En esta misma línea, se expondrán brevemente las características del esquema clásico o paramétrico y el no paramétrico para estimar series de tiempo y se discutirán sus ventajas y desventajas. Finalmente justificamos la elección de los modelos utilizados en este documento para predecir el valor de una serie de tiempo de acciones.

Fundamentos

Existen varias definiciones de series de tiempo, desde las más teóricas en donde se les observa como objetos o instancias de un proceso aleatorio dentro de un espacio de Hilbert, hasta las más simples donde se les considera como un conjunto de observaciones con una cierta interdependencia temporal. En este texto, definiremos a las series de tiempo como un conjunto de variables medidas en la misma escala e indexadas por un parámetro temporal.

El objetivo final del campo de análisis de las series de tiempo, es llegar a obtener cierto grado de conocimiento sobre estas con el fin de llevar a cabo tareas de resumen, decisión,

descripción o predicción. De manera muy general, podemos considerar dos enfoques para el estudio de las mismas, a saber, el clásico o paramétrico y el no paramétrico.

Ambos modelos poseen sus ventajas y sus desventajas. Las principales ventajas del método paramétrico es que es simple, interpretable y que no requiere de una gran cantidad de datos para poder hacer inferencia o predicción. Las ventajas de los métodos no paramétricos se concentran en que disminuyen el número de supuestos que el investigador debe incluir en el modelo y por ende deja que los datos “hablen”. La principal desventaja de los métodos no paramétricos es que tienden a ser muy intensivos tanto computacionalmente como en los requerimientos en cuanto a cantidad de datos. Antes de profundizar en ambos esquemas analizaremos un poco la estructura y componentes de una serie de tiempo.

Estructura de una serie de tiempo

Cómo se mencionó en la introducción, existen diversos tipos de series de tiempo y estas están determinadas por su comportamiento. Algunos ejemplos de series de tiempo pueden ser vistos en la siguiente Figura:

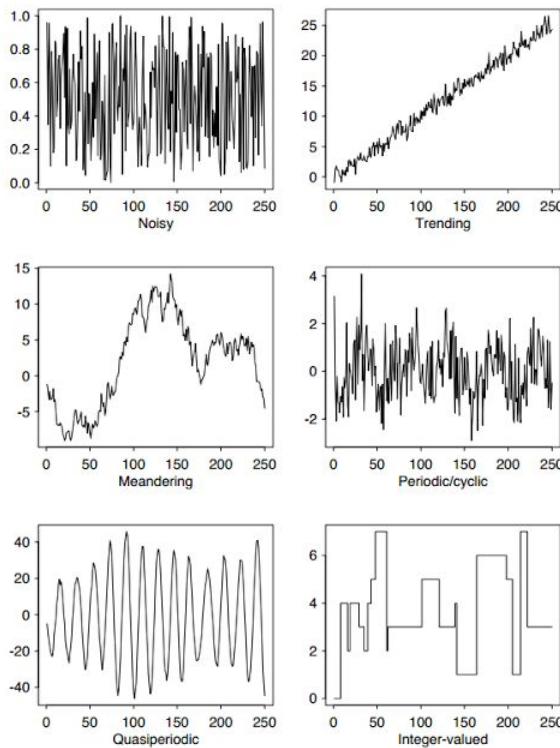


Figura 1: Distintos tipos de series de tiempo.

Claramente la primera gráfica es muy diferente de la segunda y esta del resto de las series. Podemos ver que hay algunas cuyo valor sería mucho más fácil predecir que el de otras. Esta noción de simplicidad resulta de una combinación de diversos factores tales como la varianza, estacionalidad, ciclicidad y tendencia.

Planteamiento Clásico

En este sentido, los métodos clásicos, en particular el modelo clásico aditivo, de modelado de series de tiempo, definen al valor x_t de una serie de tiempo cómo:

$$x_t = T_t + S_t + a_t$$

Donde T_t representa la tendencia o comportamiento a largo plazo de la serie, S_t representa el componente estacional y a_t representa ruido aleatorio, en la Figura 2 se puede observar la descomposición de una serie de tiempo en sus diversos componentes. Este modelo es adecuado cuando para representar series de tiempo cuya variabilidad permanece constante conforme al avance del tiempo. De aquí, el problema de modelado de una serie de tiempo se limita a aislar por medio de diversas metodologías el componente estacional y temporal del ruido. Para llevar a cabo esta tarea, se puede recurrir a dos enfoques, el paramétrico y el no paramétrico. A continuación haremos un breve recuento de los supuestos y características principales de cada método así como de sus ventajas y desventajas.

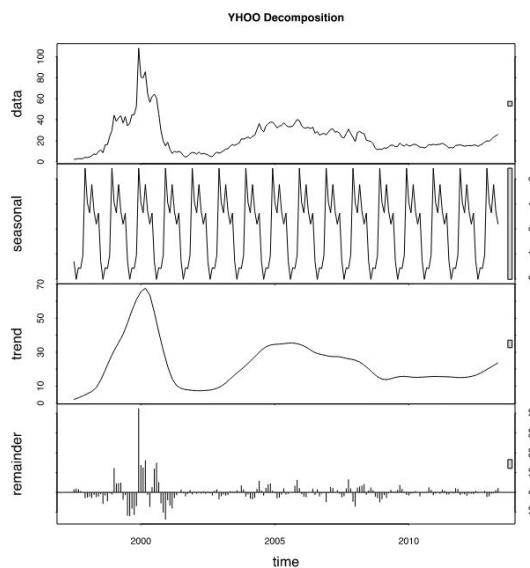


Figura 2: Descomposición de una serie temporal.

Métodos paramétricos

Propone diversas formas funcionales de variables aleatorias para expresar la relación que la tendencia guarda con el componente estacional a lo largo del tiempo. Esta relación funcional está gobernada por ciertos parámetros los cuales serán determinados por medio de un proceso de optimización de alguna función objetivo dada (ejemplo: la suma de los cuadrados de las discrepancias entre los valores observados y los propuestos por el modelo).

Una vez seleccionado el modelo, este se utilizará para aislar los efectos de la tendencia de los de la componente estacional.

Métodos no paramétricos

Los métodos no paramétricos asumen la existencia de cierta “suavidad” en la relación temporal existente entre los valores observados en el tiempo t con aquellos observados en el tiempo $t-k$. Para aproximar dicha función “suave” se pueden utilizar diversas técnicas como promedios móviles o promedios móviles ponderados. Con tal de dar diferentes niveles de importancia a diversos periodos de tiempo también se han implementado metodologías que hacen uso de kernels Mukherjee[7].

Por considerar un número menor de supuestos, en este documento nos enfocaremos en los métodos no paramétricos. En la siguiente sección profundizaremos más en sus componentes teóricos y haremos un breve recuento de las principales técnicas utilizadas para ajustarlos.

Teoría de Métodos no Paramétricos

El uso de técnicas no paramétricas para modelar series de tiempo tiene una larga tradición en el campo de la estadística. Esto es tal que el inicio del análisis espectral de una serie puede remontarse a 1898 donde Schuster introdujo el uso del periodograma Härdle[8]. A continuación expondremos algunos de los modelos no paramétricos más utilizados en la actualidad. Una de las principales ventajas de este tipo de análisis es que presentan propiedades de convergencia conforme el número de muestras aumenta.

En el resto de esta sección denotaremos por X_t a una instancia de una serie de tiempo y por $\{X_t\}$ a la serie de tiempo en conjunto.

Análisis Espectral

Supongamos que la serie de tiempo $\{X_t\}$ es univariada, estacionaria, con media cero. supongamos además que tiene autocovarianza $E(X_t, X_{t+k})=\gamma_k$. Entonces, la densidad espectral de $\{X_t\}$ está dada por:

$$f_x(w) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma_k e^{-ikw} = \frac{1}{2\pi} (\gamma_0 + 2 \sum_{k=1}^{\infty} \gamma_k \cos(kw))$$

En este sentido, la descomposición espectral de una serie puede ser vista como una suma ponderada de las componentes cíclicas correspondientes a las frecuencias w en el intervalo $[-\pi, \pi]$

Estimación de la Media Condicional

1) Media y mediana locales

Consideremos el modelo lineal general autorregresivo de orden p.

$$X_t = f(X_{t-1}, \dots, X_{t-p}) + \epsilon_t$$

y definamos : $Y_t = (X_{t-1}, \dots, X_{t-p})$ entonces, si hacemos $I_n(y) = \{i: 1 < i < n, \|Y_i - y\| < \delta_n\}$ para $\delta_n > 0$ definimos al estimador por medias como:

$$\hat{f}(x_1, \dots, x_p) = \hat{f}_n(y) = N_n(y)^{-1} \sum_{i \in I_n(y)} X_i; \quad N_n(y) = \#I_n(y)$$

El estimador por mediana se define de manera análoga.

2) Estimación por kernel

Cómo mencionamos en la sección anterior, el método de kernels es ampliamente utilizado en la estimación de series de tiempo. Para estimar la media condicional de una serie de tiempo, se utiliza la siguiente función.

$$\hat{f}(x_1, \dots, x_p) = \frac{\sum_{t=p+1}^n \prod_{i=1}^p K\left\{\frac{x_i - X_{i-1}}{h_i}\right\} X_t}{\sum_{t=p+1}^n \prod_{i=1}^p K\left\{\frac{x_i - X_{i-1}}{h_i}\right\}}$$

Donde K es una función kernel con soporte acotado.

3) Regresión polinomial local

La estimación por medio de ajuste de polinomios locales es otra alternativa al cálculo de medias condicionales. En este método, se utilizan polinomios de un grado especificado previamente y se utilizan los miembros de una vecindad de cada punto para hacer la estimación. La fórmula se obtiene de solucionar un problema de optimización cuadrático en donde se le da un mayor peso a las observaciones cercanas. El argumento final es el siguiente:

$$c_n(x) = \operatorname{argmin}_{x \in R} \sum_{t=1}^n (X_t - c^T U_m)^2 K\left(\frac{X_{t-1} - x}{h}\right)$$

donde

$$U_m = F(u_m),$$

$$F(u) = \left(\frac{l, u, \dots, u^{l-1}}{(l-1)!} \right)^T,$$

$$u_m = \frac{X_{t-1} - x}{h}$$

Modelos lineales

Finalmente, existen otro tipo de métodos que expresan la dependencia de la serie de sus instancias pasadas por medio de la composición lineal de funciones dadas. Un ejemplo de este esquema son las redes neuronales.

1) Redes Neuronales

Las redes neuronales han sido utilizadas en diversos contextos donde se requiere estimar funciones muy complejas. En este texto utilizaremos exclusivamente redes neuronales de una sola capa ya que estas, según probó Cybenko[9] pueden aproximar cualquier función continua, acotada. Bajo este contexto, podemos escribir al estimador de la serie de tiempo como:

$$f_n(X_{t-1}, \dots, X_{t-p}) = \beta_0 + \sum_{j=1}^q G(\gamma_0 + Y_t^T \gamma_j) \beta_j$$

En las siguientes secciones llevaremos a cabo la implementación de una red neuronal y un modelo polinomial entrenado con algoritmos genéticos para predecir el comportamiento de una serie de tiempo que contiene el precio de acciones.

Proyecciones en el tiempo (estructura de las BDs)

A diferencia de las series de corte transversal (series con información recabada en un mismo momento), las series de tiempo poseen una estructura bien definida con observaciones ordenadas (en el tiempo) y equidistantes entre sí (para las adyacentes).

A diferencia de los enfoque tradicional utilizado para hacer el análisis y pronósticos de las series de tiempo con modelos ARIMA, ARCH, GARH, etc..., el uso de redes neuronales, o modelos polinomiales con algoritmos genéticos, requiere de un tratamiento especial de la información.

En cualquiera de estos casos resulta necesario el escalamiento de las variables al intervalo [0,1] y la eliminación de las k observaciones que se utilizaran en las series del modelo. Por ejemplo, en un modelo como el siguiente:

$$X_t = f(X_{t-k} + Y_{t-k} + \dots + Z_{t-k})$$

Si la base original cuenta con n tuplas (formadas por las observaciones de las series $X_{t-k} + Y_{t-k} + \dots + Z_{t-k}$), la base con la que se deberá trabajar contará con n-k tuplas.

| Tiempo | X_t | X_{t-k} | Y_{t-k} | ... | Z_{t-k} |
|-----------|----------|-----------|-----------|-----|-----------|
| n | 0.299343 | — | — | ... | — |
| n - 1 | 0.132157 | — | — | ... | — |
| ... | ... | ... | ... | ... | ... |
| n - k + 1 | 0.557510 | — | — | ... | — |
| n - k | 0.341214 | 0.341214 | 0.201283 | ... | 0.010746 |
| ... | ... | ... | ... | ... | ... |
| 2 | 0.603768 | 0.018838 | 0.109009 | ... | 0.202979 |
| 1 | 0.900591 | 0.260415 | 0.680277 | ... | 0.685437 |

Elección de la base de datos

Las series que fueron elegidas para el presente análisis son el precio diario de cierre de las acciones de las 6 compañías más bursátiles (más negociadas) durante el último año en el NYSE (New York Stock Exchange). Éstas resultaron ser:

1. *BAC*

Bank of America. Corporación de servicios financieros.

2. *SUNE*

SunEdison. Líder mundial en desarrollo de energías renovables.

3. *GE*

General Electric. Corporación de compañías principalmente en las áreas de tecnología, infraestructura, servicios financieros, y medios de comunicación.

4. *PFE*

Pfizer. Laboratorio farmacéutico enfocada a la investigación, desarrollo y comercialización de medicamentos.

5. *PBR*

Petrobras. Petrolera brasileña semi-pública con participación estatal mayoritaria.

6. *SPRINT*

Empresa Norteamericana de Telecomunicaciones.

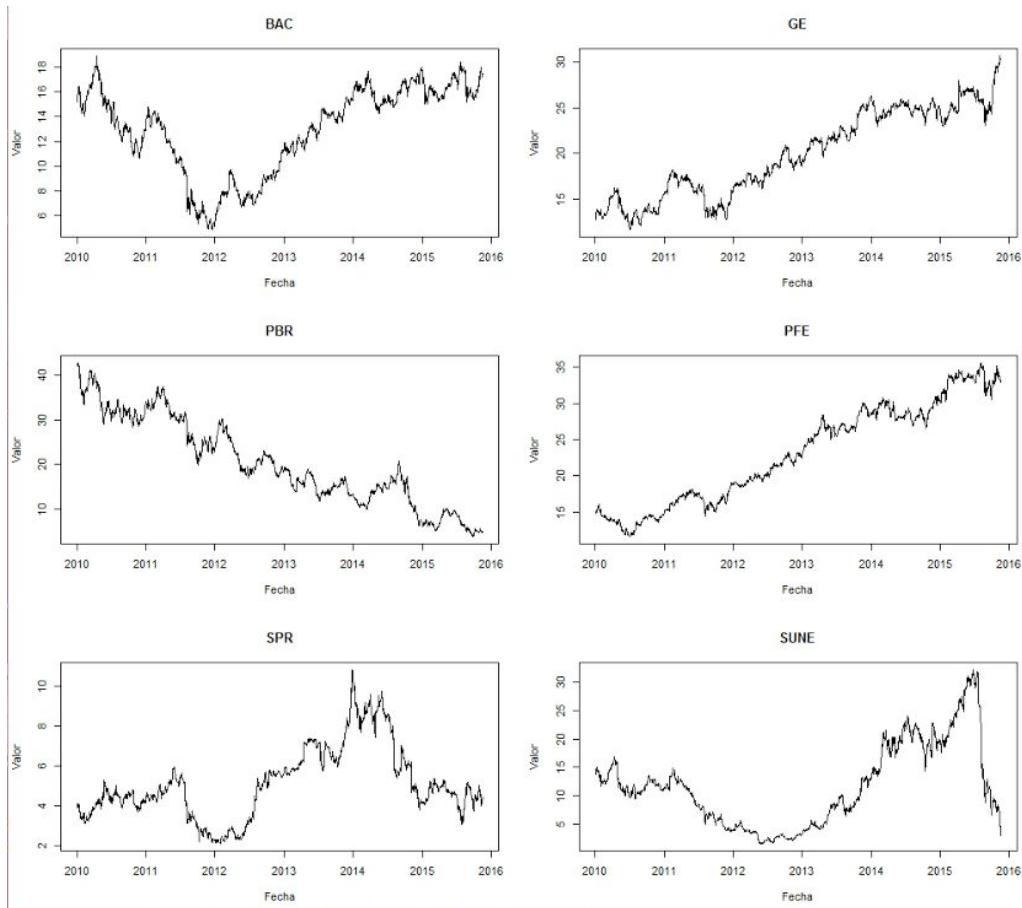


Figura 3: Gráficas de las series de tiempo para las seis acciones seleccionadas.

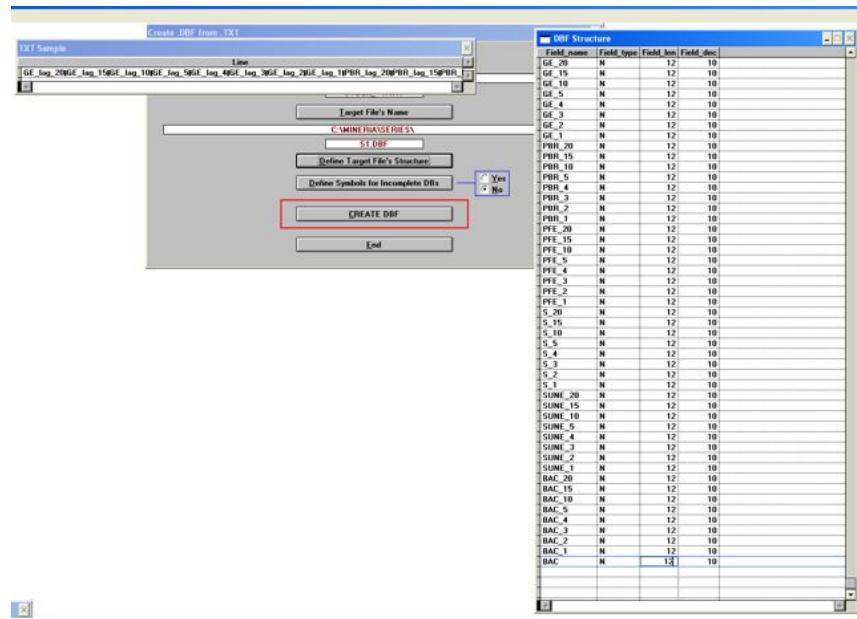
Se cuenta con la información diaria (días hábiles de USA) en el periodo enero 2010 a noviembre de 2015. Primeramente se procedió a analizar estas series y posteriormente se procedió a su preprocesamiento.

| FECHA | BAC | GE | |
|---------------------|----------------|----------------|-----------------|
| Min. : 2010-01-04 | Min. : 4.87 | Min. : 11.60 | |
| 1st Qu.: 2011-06-21 | 1st Qu.: 10.42 | 1st Qu.: 15.82 | |
| Median : 2012-12-10 | Median : 13.84 | Median : 19.74 | |
| Mean : 2012-12-10 | Mean : 12.93 | Mean : 19.95 | |
| 3rd Qu.: 2014-06-02 | 3rd Qu.: 15.87 | 3rd Qu.: 24.61 | |
| Max. : 2015-11-17 | Max. : 18.88 | Max. : 30.67 | |
| PBR | PFE | SPR | |
| SUNE | | | |
| Min. : 3.72 | Min. : 11.56 | Min. : 2.120 | Min. : 1.540 |
| 1st Qu.: 13.27 | 1st Qu.: 16.65 | 1st Qu.: 4.048 | 1st Qu.: 5.247 |
| Median : 18.47 | Median : 22.95 | Median : 4.800 | Median : 10.985 |
| Mean : 20.35 | Mean : 23.12 | Mean : 5.133 | Mean : 11.690 |
| 3rd Qu.: 29.84 | 3rd Qu.: 28.87 | 3rd Qu.: 6.008 | 3rd Qu.: 16.008 |
| Max. : 42.77 | Max. : 35.58 | Max. : 10.790 | Max. : 32.130 |

Preprocesamiento

Se llevó a cabo el preprocesamiento con el programa PREPROC.

Como se comentó, primeramente se procedió a escalar las variables. los “Lags” elegidos para probar fueron 1, 2, 3, 4, 5 (semana hábil), 10 (quincena hábil), 15, 20 (aproximadamente un mes hábil).



Resultando entonces la siguiente base:

Ahora se procede a calcular la matriz de correlaciones.

Y se identifican las correlaciones mayores al 80%

Con esta información se procede a elegir las variables que utilizaremos en el modelo.

Modelos de variables

En el aprendizaje de máquina y estadísticas la selección de variables es el proceso de selección de un subconjunto de variables relevantes (predictoras, explicativas) para su uso en la construcción del modelo. Las técnicas de selección de variables son usadas por tres razones:

- Simplificación de los modelos para que sean más fáciles de interpretar.
 - Los tiempos en los que se entrena los modelos sean más cortos.
 - Reforzar la generalización mediante la reducción del sobreajuste.

La muy conocida “Maldición de la dimensionalidad” se refiere precisamente a que mientras más dimensiones (atributos) tengan los datos, más ejemplos de entrenamiento se requerirán para aprender. La complejidad de un modelo muchas veces depende del número de atributos en los datos, por lo que el reducir el número de atributos reduce la complejidad del modelo.

Los métodos que nos ayudan a reducir la dimensionalidad se dividen en dos:

- Selección de atributos
 - Extracción de rasgos

La idea central de utilizar una técnica de selección de variables es que los datos contienen muchas características que son ya sea redundantes o irrelevantes.

Las técnicas de selección de variables deben distinguirse de las de extracción de variables. La extracción de variables crea nuevas variables por medio de transformaciones (funciones) de las variables originales. En cambio la selección de variables devuelve un subconjunto de las mismas. Las técnicas de selección de variables son usadas muy a menudo en casos donde hay muchas variables y, en forma relativa, muy pocos datos.

Ejemplos de técnicas de selección de atributos:

- Selección de subconjuntos
- Correlaciones

Ejemplos de técnicas de extracción de rasgos:

- Análisis de componentes principales

Selección de atributos

Selección de subconjuntos

La selección de subconjuntos evalúa un subconjunto de características como un grupo para determinar su idoneidad.

Muchos enfoques de búsqueda convencionales evalúan de forma iterativa un subconjunto de variables, modifica el subconjunto y evalúa si el nuevo subconjunto es una mejora sobre el anterior. El hecho de pasar por todos los subconjuntos posibles es impráctico dadas todas las combinaciones de los atributos con los que se cuentan. Por lo tanto, en estos casos se toma un tiempo de parada definido y el subconjunto de características con la puntuación más alta descubierto hasta ese momento es el que se selecciona como el subconjunto de variables satisfactorio. El criterio de parada varía según el algoritmo; posibles criterios incluyen: que la puntuación (de acuerdo a una métrica ya establecida) del subconjunto exceda a cierto umbral, o que el tiempo máximo permitido de ejecución del programa haya sido superado, etc.

Estas técnicas se pueden dividir en dos:

- Envolturas
- Filtros

Envolturas

Utilizan un algoritmo que busca a través del espacio de las posibles características y evaluar cada subconjunto mediante la ejecución de un modelo en el subconjunto.

La desventaja es que esta técnica es computacionalmente cara y tiene un riesgo importante de caer en un sobreajuste de los datos

Filtros (correlaciones)

Son similares a las envolturas en la forma en la que realizan la búsqueda, pero en lugar de evaluar con respecto a un modelo, se evalúa un filtro simple.

Una de las más simples y más poderosas técnicas de filtros es el uso de matriz de correlaciones. En problemas de regresión y de minería de datos, suele suceder que algunas variables estén altamente correlacionadas con alguna(s) otra(s) y acaben siendo redundantes.

Un buen subconjunto es aquél que contiene variables altamente correlacionadas con la variable dependiente, pero no-correlacionada entre sí mismas. La medida de ajuste que maneja debería de desechar variables irrelevantes ya que serían pobres predictivas de la variable respuesta.

Las variables redundantes deberían de ser ignoradas ya que éstas están correlacionadas con una o más de las otras variables.

En comparación con la técnica de la Envoltura, sucede que el de Filtros es mucho más rápido para casos con muchas variables explicativas más factibles.

Extracción de rasgos

Análisis de Componentes Principales

El análisis de componentes principales es una herramienta estadístico-matemática de gran utilidad en multitud de campos, especialmente en el de la comprensión de datos.

Esta técnica convierte un conjunto de observaciones de rasgos posiblemente correlacionadas en un conjunto de valores que pertenecen a nuevas variables que no están correlacionadas linealmente entre sí y que explican la variación de los datos. Estas nuevas variables reciben el nombre de Componentes Principales y cada una de éstas son combinaciones lineales de las variables originales.

El objetivo del análisis de componentes principales es la reducción de la dimensión en el conjunto de datos de entrada, reteniendo tanta varianza como sea posible. Esta reducción de dimensión ayuda a eliminar la información redundante y el “ruido” de los datos, y se lleva a cabo mediante la transformación lineal que se mencionó anteriormente, la cual disminuye el

número de variables hasta un número deseado, de modo que se mantenga máxima la cantidad de varianzas en los datos de entrada

Los componentes se generan de tal manera que el primer componente es el que tiene la mayor varianza posible y las subsecuentes estarán ordenadas decrecientemente con respecto a la cantidad de varianza que contienen. A partir de allí, se opta por algunos de los criterios que ya existen para elegir el número de componentes a utilizar.

Uno es el criterio del codo el cual consiste en graficar las varianzas de cada componente y ver a partir de cual componente la pendiente provoca tal cambio que deja ver la formación de un “codo” en la gráfica. Bajo ese criterio, se utilizan los componentes anteriores.

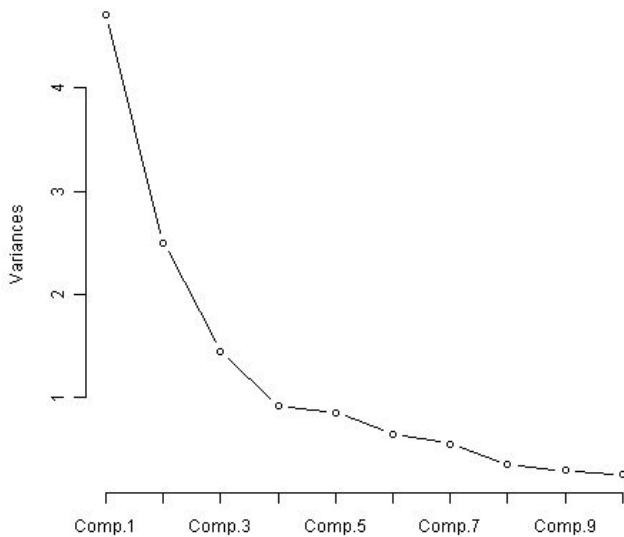


Figura 4. Varianza por componente

En el ejemplo de arriba (Figura 4) tenemos un problema de 10 componentes ordenados del primero (el que contiene la mayor varianza) al décimo (el que contiene la menor varianza). En la gráfica podemos observar que el codo se forma en el cuarto componente, por lo que se tomarían los componentes 1,2 y 3.

Otra forma de decidir cuántos componentes retener es usando el criterio de kaiser, bajo el cual deberíamos retener sólo los componentes principales para los cuales, la varianza es mayor a 1.

En el ejemplo que tenemos en la gráfica, vemos que los primeros tres componentes son los únicos que cumplen esa condición, por lo que también tomaríamos sólo a esos tres bajo este criterio.

Un tercer criterio para decidir cuántos componentes principales utilizar es simplemente tomar los que expliquen al menos un mínimo (previamente establecido) de la varianza total. Por ejemplo si el mínimo estipulado para el mismo caso hubiese sido 80%, tendríamos que tomar

los primeros cinco componentes ya que con esos se explica un 80.2% de la varianza (mientras que con los primeros cuatro explicamos sólo un 73.6% de la varianza y eso aún no es suficiente dado el mínimo establecido).

Por lo tanto, la reducción, cuando es posible, se logra ya que el número de componentes principales elegidos será menor al número de rasgos o atributos originales.

El enfoque para series de tiempo

Ahora bien, una vez que tenemos una idea general de las principales técnicas para la selección y extracción de variables, podemos ver cuál o cuáles de esas se adaptan más a lo que necesitamos dado el enfoque que estamos tratando de series de tiempo.

Por lo general, para series de tiempo cuando establezcamos un conjunto de variables inicial o “completo” muchas veces estará conformado por variables “lag” o de retraso que no son otra cosa más que los valores de un mismo atributo pero que sucedió una “t” cantidad de períodos atrás.

Luego entonces, si tenemos varios de esos para nuestros atributos es lógico pensar que muy probablemente tendremos variables que se encuentren correlacionadas dada la naturaleza de las mismas. Dicho lo anterior, para reducir la dimensionalidad de nuestro modelo necesitaríamos saber cuáles de las variables que tenemos son redundantes en vez de generar transformaciones de ellas y a partir de seleccionar un subconjunto. Dada esta situación la técnica que mejor se adapta al enfoque que estamos tratando es la de realizar un filtro por medio de una matriz de correlaciones.

Modelos de redes neuronales para la predicción de series temporales

Motivación

Las redes neuronales son aproximadores universales de funciones que pueden aproximar con cualquier grado de precisión cualquier función continua no-lineal. Su flexibilidad como aproximadores de cualquier tipo de función las convierten en un método poderoso para el reconocimiento de patrones, clasificación y la realización de pronósticos.

Las redes neuronales pueden ser utilizadas para realizar pronósticos sobre series temporales con la ventaja de que las dependencias entre las variables se aprenden de manera automática sin necesidad de añadir información adicional (como lo es el tipo de dependencia en el caso de modelos de regresión polinomial). Una red neuronal entrenada sobre datos históricos puede

descubrir las dependencias escondidas en las variables, inclusive si no se tiene conocimiento a-priori sobre las relaciones entre las variables y/o se desconoce la estructura de autocorrelación de la serie, para predecir con precisión el valor (regresión) o tendencia (clasificación) de una variable en el futuro. Sin embargo, es importante tomar en cuenta que el uso de redes neuronales tiene sus desventajas.

Las soluciones propuestas tienden a ser cajas negras de difícil interpretación, tienen tiempos de entrenamiento largos, el peligro de sobreentrenamiento y la gran cantidad de parámetros que deben ser seleccionados experimentalmente para obtener buenos pronósticos.

Debido a las ventajas mencionadas, las redes neuronales han ganado popularidad en el campo de las finanzas. Algunas de las aplicaciones más usuales incluyen la calificación de riesgo hipotecario y de inversiones de ingreso fijo, la construcción de índices, simulaciones del comportamiento de mercados, elección y diversificación de portafolios y pronósticos económicos.

Existen numerosas técnicas estadísticas para el análisis de series de tiempo, tanto con el uso de modelos lineales como no lineales, sin embargo muchas de estas técnicas producen bajas tasas de éxito para el pronóstico de los mercados financieros. El comportamiento de los mercados financieros es altamente no-lineal debido a los muchos factores que interactúan, incluyendo eventos políticos, la condición general de la economía y las expectativas de los agentes bursátiles[3].

Caso de estudio - predicción del valor del mercado bursátil

Nos proponemos ejemplificar cómo el problema de predecir el valor de una acción bursátil puede ser resuelto de manera eficaz y precisa con el uso de una red progresiva no-recurrente de perceptrones multicapa entrenada con el algoritmo de retropropagación con información histórica de la acción en el pasado y de otras acciones bursátiles de otros sectores de peso en la economía de los Estados Unidos. Existe un extenso campo de investigación respecto a determinar qué sectores de la economía de un país empujan el desarrollo económico (construcción, energía, financiero), cuáles son relativamente independientes (farmacéutico) y cuáles en general siguen el movimiento de otros sectores (consumo, industria). Decidimos analizar y predecir el movimiento del valor de las acciones de Bank of America en función a sus valores históricos y a los de las acciones en los sectores energético (PETROBRAS - PBR y SunEdison - SUNE), farmacéutico (Pfizer - PFE), industrial (General Electric) y el de las telecomunicaciones (Sprint Telecom - SPR). La literatura sugiere que existe una gran cantidad de factores que afectan los valores de las acciones en la bolsa y sugieren que los precios de las acciones capturan las señales del mercado, es posible en teoría predecir el valor de las acciones con su comportamiento histórico y el de otros índices o valores que cambian con el tiempo (como por ejemplo el valor de otras acciones como en nuestro ejercicio).

De manera general entonces nuestro modelo asume que todos los factores que inciden en el valor de las acciones de Bank of America están capturados ya en los valores históricos de sus acciones y el de las otras 5 acciones más bursátiles del mercado de los Estados Unidos. Este puede escribirse de la siguiente manera:

$$BAC_t = f(BAC_{t-k_1} + \dots + BAC_{t-k_n} + GE_{t-k_1} + \dots + GE_{t-k_n} + PBR_{t-k_1} + \dots + PBR_{t-k_n} + PFE_{t-k_1} + \dots + PFE_{t-k_n} + SPR_{t-k_1} + \dots + SPR_{t-k_n} + SUNE_{t-k_1} + \dots + SUNE_{t-k_n})$$

Donde k representa el retraso o ventana de tiempo que tiene un efecto sobre el valor presente de la variable, es posible que la misma variable tenga un efecto con diferentes retrasos sobre la variable dependiente. Por ejemplo, las ventas mensuales de un producto se pueden predecir en función de las ventas del mes anterior y de las ventas del mismo periodo del año anterior:

$$Ventas_t = f(Ventas_{t-1} + Ventas_{t-12})$$

La elección de las ventanas de retraso k_1 a k_n dependen mucho del problema específico que se quiere resolver. Una posibilidad para determinar estas ventanas de retraso sería el uso de técnicas de selección de atributos para determinar qué ventana o ventanas de retraso son las que tienen mayor relación con la variable dependiente y utilizar solo estas para la predicción.

Construcción de las variables con ventana de retraso

De acuerdo a la literatura, en el sector financiero existe una fuerte volatilidad diaria en el valor de las acciones por lo que las ventanas de retraso grandes no tienen impacto en los valores futuros de las acciones. Para efectos de nuestro ejercicio tomamos 8 ventanas de retraso correspondientes a los cinco últimos días hábiles, y a las 4 semanas más recientes ($k \in \{1, 2, 3, 4, 5, 10, 15, 20\}$ días hábiles). Dado que se necesitan utilizar observaciones anteriores para la generación de las variables con retraso, es inevitable perder ciertas observaciones en este proceso, el número de observaciones que se pierden en este proceso corresponde a la ventana de retraso más grande. En nuestro caso este proceso generó la pérdida de 20 observaciones dado que $k_{max} = 20$. Por cada ventana de retraso calculada se genera un atributo o variable adicional en nuestra matriz de diseño por lo que después de generar las 8 variables con retraso para nuestras 6 acciones tenemos una matriz de diseño X con $(8 \times 6) = 48$ predictores y $(n - k_{max}) = 1480 - 20 = 1460$ observaciones.

Selección de atributos

Es importante recordar que la naturaleza misma de las series de tiempo hace que un gran número de nuestras variables con retraso están fuertemente relacionadas entre sí, esta relación se da por la misma estructura de autocorrelación temporal subyacente en los datos. Debido a esto la selección de atributos es un punto particularmente importante para el análisis de series de tiempo para evitar que los algoritmos que vamos a utilizar funcionen

adecuadamente. Para nuestro ejercicio decidimos utilizar la técnica de análisis de correlación de Pearson que hemos utilizado en clase, y les daremos también una demostración de la opción de podado (pruning) de la red neuronal que el software Data Engine ofrece como parte de su funcionalidad como un ejemplo de selección de atributos embebido en el entrenamiento. Aunque existen otras técnicas más sofisticadas quisimos mantener al mínimo el nivel de pre-procesamiento, transformación y ingeniería de atributos en este ejercicio para demostrar cómo las redes neuronales pueden dar buenos resultados sin el uso de técnicas de pre-análisis de mayor sofisticación. Tomamos tres valores umbral para la extracción de las variables correspondientes al 80%, 90% y 95%. Los resultados de este proceso se muestran a continuación:

| Umbral de índice de correlación | Variables con retraso seleccionadas |
|---------------------------------|---|
| 80% | PFE _{t-1} , SPR _{t-1} , SUNE _{t-1} , BAC _{t-1} |
| 90% | PBR _{t-1} , PFE _{t-1} , SPR _{t-1} , SUNE _{t-1} , BAC _{t-1} |
| 95% | GE _{t-1} , PBR _{t-1} , PFE _{t-1} , SPR _{t-1} , SUNE _{t-1} , BAC _{t-1} |

Se decidió tomar el umbral de 95% para mantener al menos una variable con retraso de cada uno de los sectores que seleccionamos. Es importante mencionar que la elección del retraso = 1 tiene que ver con el ordenamiento de las columnas en la matriz de diseño. De haber ordenado al revés la matriz el algoritmo de extracción hubiera mantenido las columnas para un retraso de k=20. Por este motivo decidimos probar 8 modelos diferentes con cada uno de los retrasos que pre-seleccionamos para escoger aquel que de las predicciones más acertadas.

En otras palabras nuestro modelo de predicción se puede resumir como sigue:

$$BAC_t = f(BAC_{t-k} + GE_{t-k} + PBR_{t-k} + PFE_{t-k} + SPR_{t-k} + SUNE_{t-k})$$

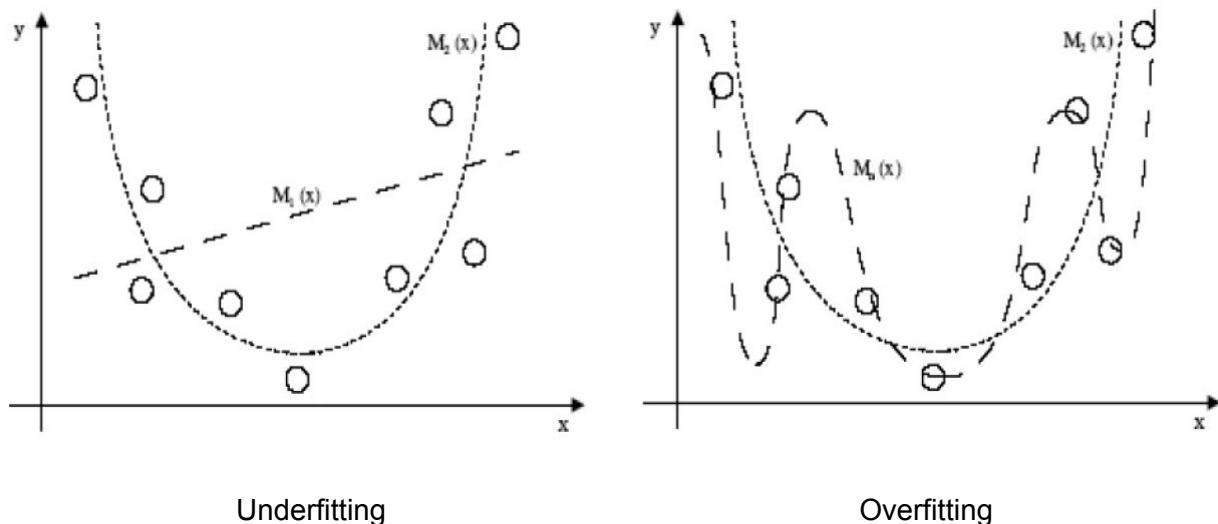
Donde $k \in \{1, 2, 3, 4, 5, 10, 15, 20\}$ para cada uno de los 8 modelos (nótese que la ventana de retraso es siempre la misma para las seis variables independientes en cada uno de los 9 modelos).

Selección del tipo de red neuronal, su arquitectura y los parámetros de entrenamiento

Una de las grandes ventajas de las redes neuronales es su flexibilidad para aproximar cualquier función continua, sin embargo esta flexibilidad tiene que ver con la gran cantidad de parámetros que deben de seleccionarse sin reglas exactas y requieren de una ardua labor de

exploración por parte del investigador. Para efectos de este ejercicio tomamos la decisión de basarnos en un ejercicio previo de aprendizaje con redes neuronales aplicado a la epidemiología listado en las referencias[1] y al tutorial del software Data Engine para redes de perceptrones multicapa aplicado a series de tiempo: "The problem of passenger number forecasting". Decidimos utilizar una red neuronal progresiva no-recurrente con perceptrones multicapa basada en el algoritmo de retropropagación con momento siguiendo el ejemplo mencionado dado que es una de las redes más sencillas y ha sido exitosa en la predicción de series de tiempo en numerosos problemas. Utilizamos un enfoque de validación cruzada en dónde detenemos el aprendizaje de la red cuando la métrica de error sobre un conjunto de prueba alcanza su valor mínimo, a esta estrategia de entrenamiento se le conoce como "early stop".

La arquitectura de la red neuronal está dada en gran medida por las variables de entrada y de salida. Como hemos mencionado anteriormente, está demostrado que una sola capa intermedia es suficiente para aproximar cualquier función continua. Determinar el número de neuronas en las capas intermedias es un problema complejo, demasiadas neuronas y el modelo podría sufrir de sobreentrenamiento además de que un mayor número de observaciones es necesario al incrementar el número de conexiones de la red, sin contar con el incremento de tiempo de entrenamiento de la red. Demasiadas pocas neuronas y el modelo será demasiado "simple" y no podrá capturar totalmente la estructura subyacente de los datos. A continuación se presentan ejemplos de modelos que sufren de sobreentrenamiento y de sobreentrenamiento.



En este caso tenemos 6 neuronas en la capa de entrada y 1 neurona en la capa de salida. Dado que contamos con un gran número de observaciones que cubren el dominio $[0,1]$ con bastante uniformidad únicamente es necesario utilizar una capa intermedia de neuronas. Incluimos un perceptrón adicional con entrada convencional de 1 que sirve como neurona de

umbral (Bias Neuron) similar al parámetro de intercepto utilizado en regresión lineal. A continuación presentamos un ejemplo de la arquitectura que nos proponemos utilizar, en este caso en el diagrama se incluyeron 4 neuronas en la capa intermedia aunque este valor no fue el que se utilizó al final.

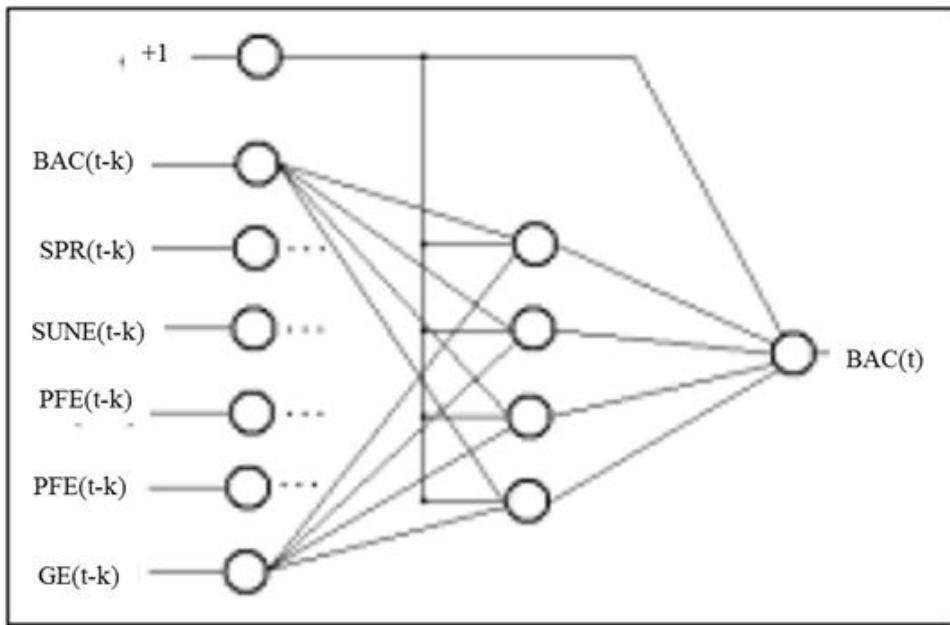


Figura 5: Ejemplificación de la arquitectura de la red neuronal con 4 neuronas intermedias (6-4-1).

Para determinar el número adecuado de neuronas en la capa intermedia para nuestros modelos utilizamos la técnica que hemos utilizado en clase basada en la idea de la entropía de la información de Shannon y el software Preproc del ITAM. Para esto se generaron nueve archivos con las 1460 observaciones y se utilizó el algoritmo de compresión de Huffman PPMZ2 sobre los nueve archivos con la información de las series de tiempo. A continuación se presentan los resultados obtenidos:

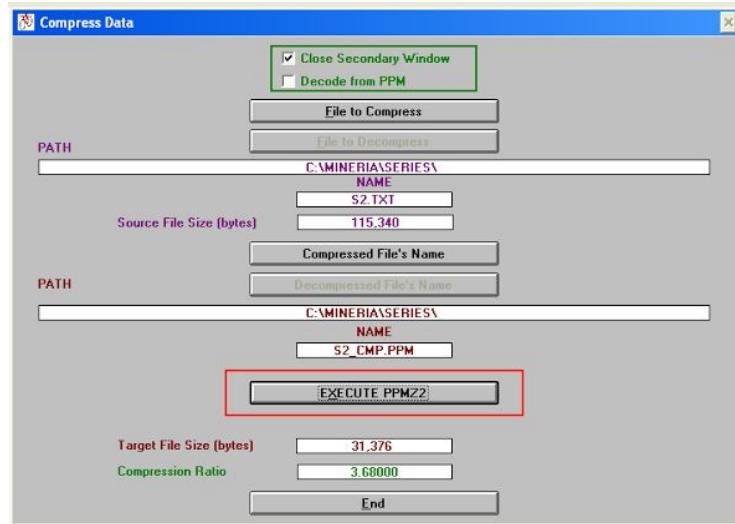


Figura 6 : Compresión obtenida para una ventana móvil de $k=2$.

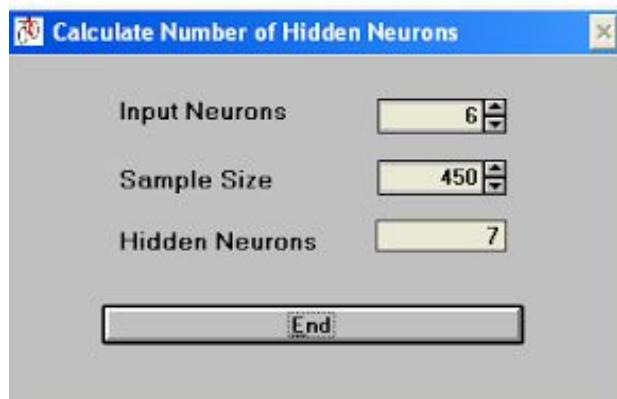


Figura 7: Ejemplo de determinación del número de neuronas de la capa intermedia para $k=20$ (Factor de compresión = $3.22 - 1460/3.22 = 450$ y 6 variables de entrada - 7 neuronas en la capa intermedia).

Los factores de compresión obtenidos fueron muy similares para los nueve archivos de datos preprocesados y en todos los casos obtuvimos que 7 neuronas deberían de ser capaces de encontrar la estructura subyacente de los datos. Se utilizó entonces una arquitectura de 6-7-1. Es decir que se tienen $6 \times 7 + 7 + 7 + 1 = 57$ conexiones o pesos.

Para la elección de los demás parámetros es importante recordar que una constante de aprendizaje demasiado grande se detecta cuando la función de error cambia rápidamente sin mostrar signos de convergencia. Una constante demasiado pequeña requiere de un largo

tiempo de entrenamiento y sufre el riesgo de caer en mínimos locales. Aunque en un problema real es importante asegurarse con la realización de pruebas de que todos los parámetros son los adecuados para efectos de este ejercicio utilizamos los parámetros propuestos en el problema de epidemiología mencionado en las referencias y las pruebas fueron exitosas (convergencia del entrenamiento en un número de épocas aceptable).

Los demás parámetros utilizados para el entrenamiento de la red fueron:

- Algoritmo de retropropagación con momentum (que permite acelerar el entrenamiento y reduce el riesgo de caer en mínimos locales).
- Las tasas de entrenamiento fueron 0.01 y 0.0001 para las capas 1 y 2.
- Las tasas de momento fueron 0.7 y 0.9 para las capas 1 y 2.
- Se utilizaron funciones de activación lineal-sigmoide-lineal para las capas en orden.
- Los pesos de inicialización son aleatorios entre -0.1 y 0.1.

Validación

Para evitar el riesgo de sobreentrenamiento de la red (aprendizaje del ruido y no de la señal en los datos) es fundamental probar los valores de los pesos de las neuronas de la red sobre un conjunto de datos de prueba que la red no utiliza en el proceso de aprendizaje. Para esto sepáramos los datos en dos conjuntos, un conjunto de entrenamiento (80%) que la red utiliza para aprender la función objetivo y un conjunto de prueba (20%) sobre el que se prueba la capacidad de generalización de la red. Cuando se trabaja con series de tiempo es fundamental que los conjuntos de entrenamiento y de prueba están compuestos de observaciones temporalmente consecutivas o se corre el riesgo de perder gran parte del poder explicativo debido a la autocorrelación temporal de los datos por lo que las técnicas de selección aleatorias simples no son adecuadas en estas situaciones. En nuestro caso, sepáramos el conjunto de 1,460 observaciones en un conjunto de entrenamiento con las primeras 1,168 observaciones consecutivas (correspondientes a los primeros 48 meses) y un conjunto de prueba con las últimas 292 observaciones (correspondientes a los últimos 12 meses). Para la obtención de mejores resultados se podría aplicar un enfoque de k-pliegues retirando del conjunto de datos el equivalente a un año de información cambiando el año extraído, sin embargo para efectos demostrativos únicamente trabajamos con un conjunto de entrenamiento y uno de prueba.

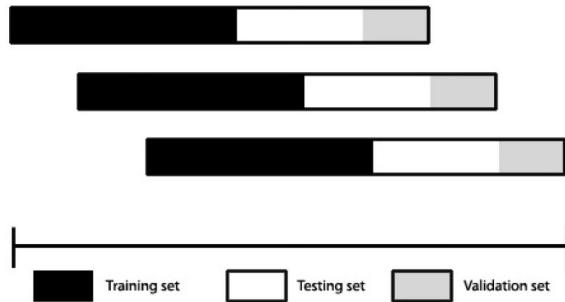


Figura 8: Estrategia de validación cruzada aplicada a series de tiempo.

La métrica de error utilizada para comparar los diferentes modelos fue la raíz del error cuadrático medio (RMSE) aunque se tomó en consideración el Error Máximo también para determinar cuándo parar el entrenamiento cuando el error en L2 era el mismo para múltiples épocas.

Entrenamiento y Resultados

Para el entrenamiento de la red neuronal se generaron 18 archivos de texto planos separados por tabulador, 2 para cada modelo, un conjunto de entrenamiento y uno de prueba. Se eliminaron los nombres de columna y se cambió el tipo de archivo a .dat. Se utilizó el modelo de redes de perceptrones multicapas del software Data Engine en su versión de estudiante 2.1 siguiendo las indicaciones del Tutorial: Redes Neuronales para Series de Tiempo: El problema del volumen de pasajeros.

Al analizar los resultados de las nueve redes neuronales entrenadas con ventanas de retraso diferentes, llevó a los siguiente resultados (únicamente se reportan los errores calculados sobre el conjunto de prueba).

| Ventana de retraso | Raíz del error cuadrático medio en el conjunto de pruebas. | Error máximo en el conjunto de pruebas. | Epoca de paro |
|--------------------|--|---|---------------|
| 1 | 0.014 | 0.059 | 900 |
| 2 | 0.019 | 0.097 | 1,300 |

| | | | |
|----|-------|-------|-------|
| 3 | 0.022 | 0.141 | 1,300 |
| 4 | 0.025 | 0.155 | 1,300 |
| 5 | 0.027 | 0,159 | 1,300 |
| 10 | 0.04 | 0.157 | 2,700 |
| 15 | 0.047 | 0.147 | 1,900 |
| 20 | 0.051 | 0.139 | 1,600 |

El modelo que brinda los mejores resultados es el que tiene una ventana de retraso de $k=1$, aunque este resultado parece trivial o evidente, no siempre se obtienen los mismos resultados, por ejemplo en la referencia sobre epidemiología en México que mencionamos se utilizan tres ventanas de rechazo de 28,56 y 84 días y el mejor modelo es el que utiliza 84 días.

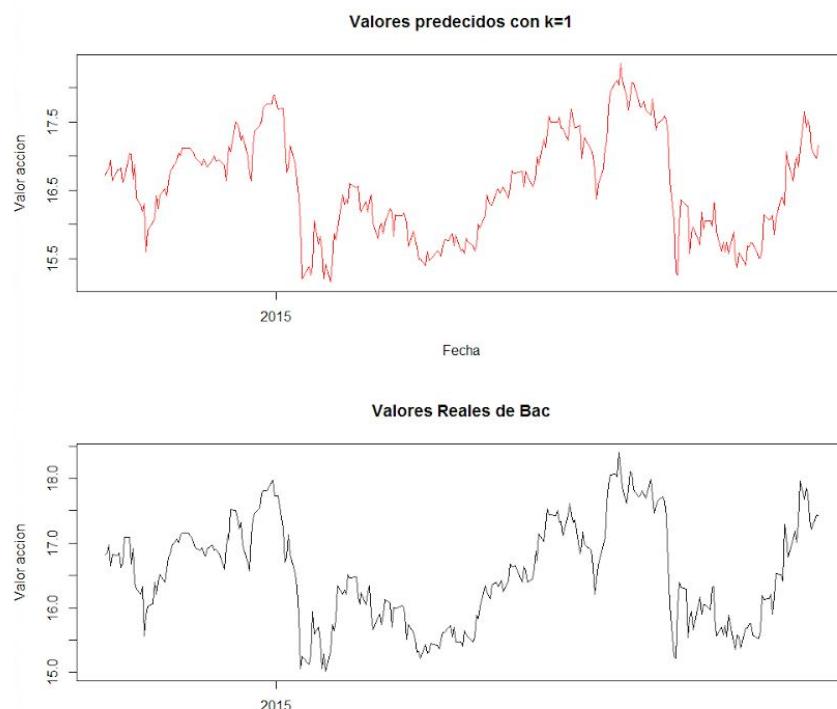


Figura 9: Valores reales de las acciones de Bank of America y valores predecidos con el modelo cuando $k=1$ día hábil.

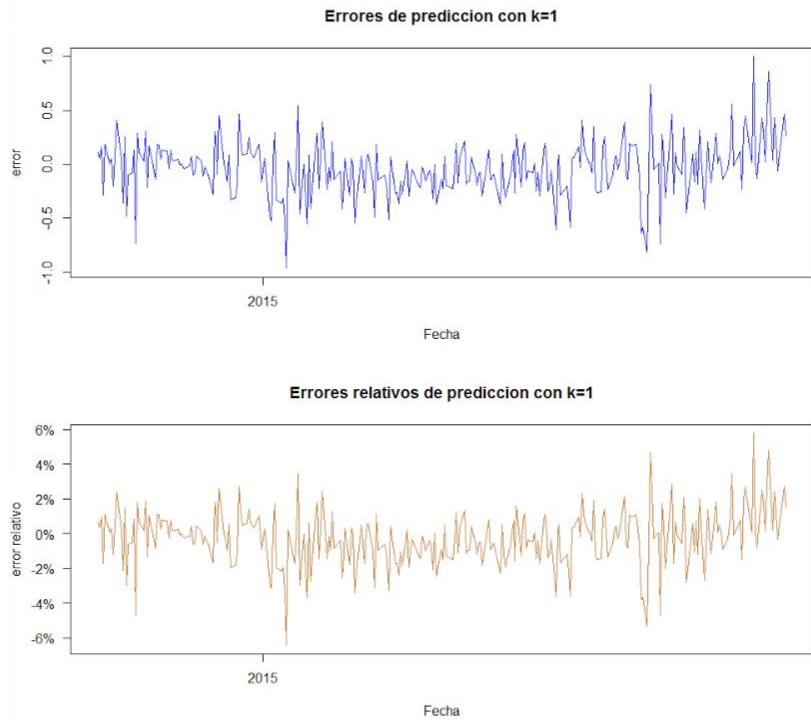


Figura 10: Errores de predicción absolutos y relativos entre el modelo $k=1$ y los valores reales de las acciones de Bank of America.

Del entrenamiento de una red neuronal se puede estimar de manera precisa (1.4% de error sobre los valores futuros de una acción, inclusive con un retraso de 5 días seguimos acertando en un rango del 2.7% al valor esperado de una acción) el valor futuro de las acciones de Bank of America, inclusive teniendo poco entendimiento del sector financiero y de la estructura de los datos. Para implementar una red neuronal para la predicción futura es necesario extraer los pesos obtenidos para el cálculo de los valores futuros. Siguiendo la notación propuesta en [1] se puede expresar el resultado del entrenamiento de la red concatenando la arquitectura utilizada (neuronas de entrada, intermedias, de salida) con las funciones de transferencia a usar (1=lineal, 2=sigmoide, etc...) y con los pesos obtenidos de cada conexión 57 valores. $BAC(k=1) = [(6,7,1);(1,2,1);(-1.168535,-0.310659,\dots,-0.363649,0.378183)]$.

Modelo con podado

Un método adicional para determinar la arquitectura óptima de la red tiene que ver con el uso de métodos adaptativos, la topología de la red se ve modificada durante el entrenamiento usando dos estrategias:

- Empezar con una red pequeña e irla creciendo conforme vaya siendo necesario.

- Empezar con una red “grande” y la va reduciendo conforme los pesos van mostrando no ser significativos en la estructura de la red. (podado).[15]

Los resultados que obtuvimos en este ejercicio pueden mejorarse de varias maneras, uno de los ejercicios adicionales que quisimos mostrarles fue el uso de la funcionalidad de podado de el software Data Engine. Esta funcionalidad permite ir eliminando conexiones que no muestran ser relevantes en el funcionamiento de la red. Sin ahondar en el funcionamiento de podado del software, básicamente se eliminan aquellos pesos que contribuyen poco a la suma total de la neurona, es decir que tienen pesos cercanos a 0 o que a pesar de ser activos tienen valores que se multiplican por 0.

Se generó un conjunto de prueba y de entrenamiento con todas las ventanas de retraso discutidas anteriormente, es decir un archivo con 48 predictores o columnas y la columna con la información a predecir. El ejercicio de decisión sobre el número de neuronas en la capa intermedia nos indicó un valor de 1. Dado que la estrategia es sobredimensionar la red y dejar que esta se reduzca se tomaron dos neuronas en la capa intermedia y se dejaron todos los demás parametros idénticos a los demás modelos. Los parametros del podado se dejaron en sus valores por defecto de 0.02 para el umbral de relevancia de las conexiones y de 10 para el número de épocas a considerar en memoria para cortar.

Los resultados son interesantes ya que después de 10,000 épocas únicamente quedan 9 variables relevantes para la predicción del valor de las acciones de Bank of America. Estas variables corresponden a GE_{t-1} , PBR_{t-15} , $SUNE_{t-2}$, BAC_{t-20} , Bac_{t-10} , BAC_{t-5} , BAC_{t-4} , BAC_{t-3} , BAC_{t-2} , BAC_{t-1} . El error RMSTest = 0.014 igual al mejor de nuestros modelos de retraso fijos y MaxErrTest = 0.062 ligeramente mayor al mejor modelo pero con un error máximo menor a los otros 7 modelos con retraso fijo.

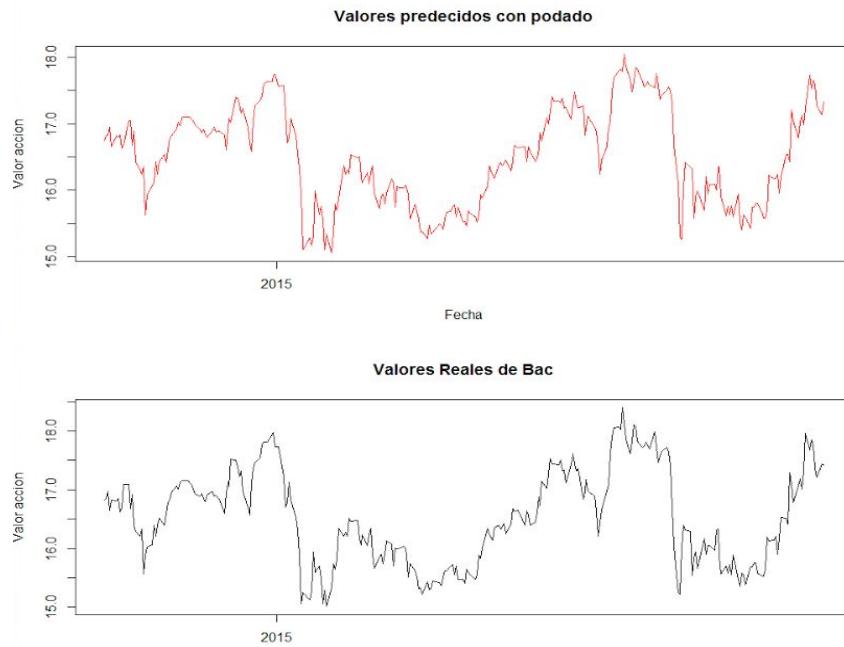


Figura 11: Valores reales de las acciones de Bank of America y valores predecidos con el modelo con GE_{t-1} , PBR_{t-15} , $SUNE_{t-2}$, BAC_{t-20} , Bac_{t-10} , BAC_{t-5} , BAC_{t-4} , BAC_{t-3} , BAC_{t-2} , BAC_{t-1} .

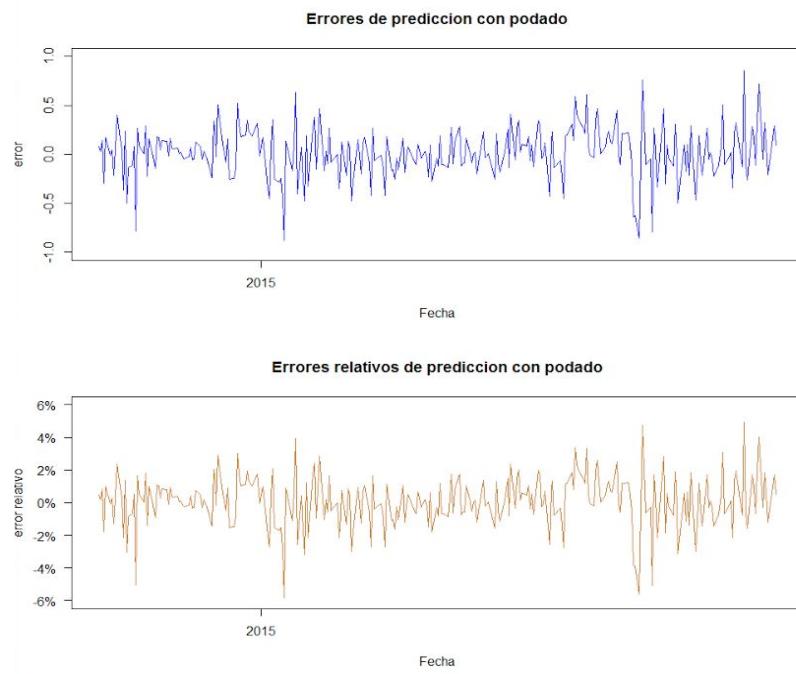


Figura 12: Errores de predicción absolutos y relativos entre el modelo con GE_{t-1} , PBR_{t-15} , $SUNE_{t-2}$, BAC_{t-20} , Bac_{t-10} , BAC_{t-5} , BAC_{t-4} , BAC_{t-3} , BAC_{t-2} , BAC_{t-1} y los valores reales de las acciones de Bank of America.

No buscamos justificar el uso de podado al trabajar con redes neuronales, pero nos pareció interesante presentarles una alternativa de selección de atributos dinámica dentro del proceso

de entrenamiento con buenas capacidades de generalización. Sin mayor preprocesamiento que el de la normalización entre 0 y 1 y sin ningún conocimiento a priori sobre el problema en cuestión hemos podido dar predicciones adecuadas al problema de predecir el precio de una acción.

Modelos polinomiales no lineales

Planteamiento del problema

En la sección anterior abordamos el problema de modelar una serie temporal usando redes neuronales. En esta sección presentamos un segundo enfoque.

Este enfoque consiste en usar una aproximación polinomial multivariada; posteriormente dicho polinomio se ajustará usando algoritmos genéticos. Este enfoque tiene la ventaja de dar una forma funcional cerrada a diferencia de los “modelos de caja negra”, como suelen ser llamados las redes neuronales, las máquinas de soporte vectorial, etc. Adicionalmente, el ajuste polinomial se puede plantear como un problema de optimización y los algoritmos genéticos son muy eficientes resolviendo este tipo de problemas.

¿Por qué usar polinomios multivariados?

Ahora bien, para entender porqué queremos modelar nuestra serie temporal usando un polinomio multivariado de la forma, es necesario introducir primero algunos resultados teóricos, los cuales fueron extraídos de *Kuri-Morales* [2].

Teorema 1. Teorema de Aproximación Universal (TAU). Sea $\varphi(\cdot)$ una función continua no constante, acotada, y monótonamente creciente. Sea I_{m_O} el hipercubo m_O -dimensional. El

espacio de funciones continuas en I_{m_O} se denota como $C(I_{m_O})$. Entonces cualquier función $f \in C(I_{m_O})$ y $\varepsilon > 0$, existe un natural M y conjuntos de constantes a_i, b_i y w_{ij} donde $i=1,2,\dots,m_I$ y $j=1,2,\dots,m_O$ tal que definimos

$$F(x_1, \dots, x_{m_O}) = \sum_{i=1}^{m_I} \left[a_i \cdot \varphi \left(\sum_{j=1}^{m_O} w_{ij} x_j + b_i \right) \right]$$

como un aproximación de la función $f(\cdot)$,

esto es, $|F(x_1, \dots, x_{m_O}) - f(x_1, \dots, x_{m_O})| < \varepsilon$ para todo x_1, \dots, x_{m_O} en el espacio de entrada.

Observación 1. El TAU nos permite aproximar cualquier función que cumpla las hipótesis del teorema.

Observación 2. El TAU se aplica de manera inmediata a redes neuronales, en específico a un perceptrón multicapa.

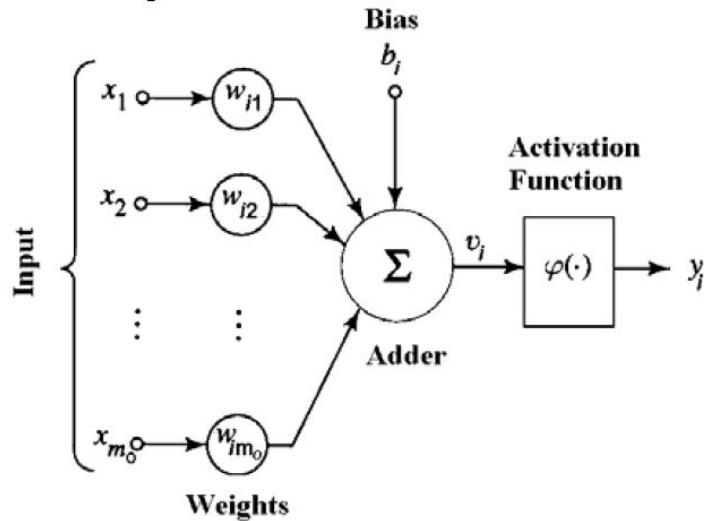


Figura 13. Perceptrón

Cuando se incluye el sesgo de la neurona de salida en $F(x_1, \dots, x_{m_0})$ del Teorema 1, obtenemos la siguiente fórmula de acuerdo a Kuri-Morales [2]:

$$F(x_1, \dots, x_{m_0}) = \alpha_0 + \sum_{i=1}^{m_I} \left[\alpha_i \cdot \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \right]$$

La ecuación del perceptrón está dada por $y_i = \varphi \left[\sum_{j=0}^{m_0} w_{ij} x_j \right]$ en donde $x_0 = 1$, $w_0 = b_0$.

Podemos elegir a $\varphi(x) = \text{logistic}(x)$ como la función no lineal de un modelo neuronal para la

construcción de un MLP. Notemos que $F(x_1, \dots, x_{m_0})$ del Teorema 1 es la salida de una MLP descrita como sigue:

- a. La red tiene m_0 nodos de entrada y una sola capa escondida que consiste de m_I neuronas. Los nodos de entradas se denotan por x_1, \dots, x_{m_0} .

- b. La neurona escondida i tiene pesos $w_{i1}, w_{i2}, \dots, w_{imO}$
- c. La salida de la red es una combinación lineal de las salidas de las neuronas escondidas con $\alpha_1, \dots, \alpha_m$ los pesos de la capa de salida.

Vemos que el TAU nos provee de la justificación matemática para aproximar casi cualquier función (se deben cumplir las condiciones del Teorema). Se tiene el siguiente corolario.

Corolario. El perceptrón multicapa requiere únicamente una capa escondida para lograr aproximar uniformemente a f en un conjunto de entrenamiento dado x_1, \dots, x_{mO} y salida objetivo $f(x_1, \dots, x_{mO})$.

Teorema 2. La función logística $1/(1+e^{-x})$ puede ser aproximada mediante un polinomio $P_n(x)$ de grado n con un error $\varepsilon = |P_n(x) - \text{logistic}(x)|$ donde $\varepsilon \rightarrow 0$ cuando n es grande

Teorema 3. Una función $y=f(x)$ en $[0,1]$ puede ser aproximada por un polinomio basado en polinomios de Chebyshev, los cuales minimizan la norma de mínimos cuadrados y la norma L_infinity.

Teorema 4. El polinomio $P_n(x)$ del Teorema 2 se puede aproximar por un polinomio $Q_l(x) = \beta_0 + \sum_{l=1}^n \beta_l x^{2l-1}$ con un error de $\varepsilon \rightarrow 0$.

Corolario 1. La función logística debe ser aproximada por el polinomio $P_{II}(x) \sim \beta_0 + \sum_{i=1}^6 \beta_i x^{2i-1}$ con un ECM de $\varepsilon_2 \sim 0.000781$ y un error mini-max $\varepsilon_\infty \sim 0.001340$, donde $\beta_0 \sim 0.5000$, $\beta_1 \sim 0.2468$, $\beta_2 \sim -0.01769$, $\beta_3 \sim 0.001085$, $\beta_4 \sim -0.000040960$, $\beta_5 \sim 0.0000008215$, $\beta_6 \sim -0.00000000664712$.

Corolario 2. Cualquier función como en 3 puede ser aproximada con una combinación lineal de polinomios que tengan una constante más términos de grado impar.

A continuación se muestra una figura obtenida de [2] ejemplificando este punto:

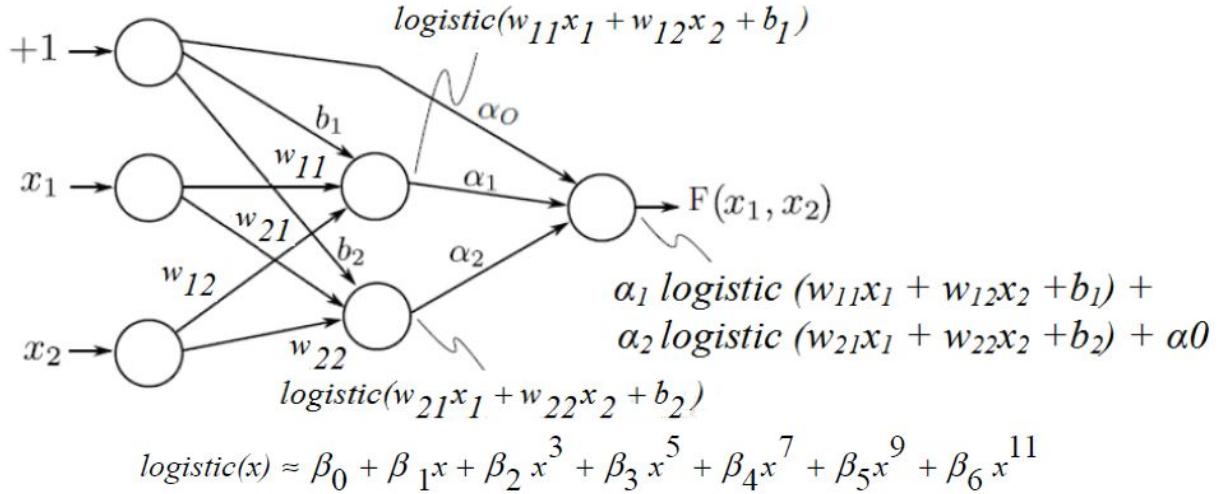


Figura 14. Polinomio resultante de usar la expansión algebraica de la función logística como la función del TAU para una función de 2 variables.

Corolario 3. En la salida de cualquier nodo oculto de un perceptrón de tres capas con m_0 nodos de input en donde hay T_0 monomios de grado menor o igual a 11, en donde $T_0 = 1 + \sum_{j=1}^6 [(2j-1+m_0)!]/[(2j-1)!m_0!]$ después de la expansión algebraica de la logística por el corolario 2.

Corolario 4. Cualquier función de m_0 variables puede ser aproximada con T monomios de grado $D \leq 121$, en donde:

$$T = 1 + \sum_{i=1}^6 \left(2i + \left(\sum_{j=1}^6 \frac{(2j-1+m_0)!}{(2j-1)!m_0!} \right) \right)! / \left((2i-1)! \left(1 + \sum_{j=1}^6 \frac{(2j-1+m_0)!}{(2j-1)!m_0!} \right)! \right)$$

Teorema 5. Teorema de aproximación universal para polinomios. Cualquier función de m_0 variables puede ser aproximada con a lo más T -términos de grado D , es decir:

$$\begin{aligned}
F(x_1, \dots, x_{m_o}) &= \alpha_0 + \sum_{i=1}^{m_I} \alpha_i \cdot \varphi \left(\sum_{k=0}^{m_O} w_{ik} x_k \right) \\
&= \alpha_0 + \sum_{i=1}^{m_I} \alpha_i \cdot \text{logistic} \left(\sum_{k=0}^{m_O} w_{ik} x_k \right) \\
&= \alpha_0 + \sum_{i=1}^{m_I} \alpha_i \cdot \sum_{j=0}^{\infty} \lambda_j \cdot \left(\sum_{k=0}^{m_O} w_{ik} x_k \right)^j \\
&= \alpha_0 + \sum_{i=1}^{m_I} \alpha_i \cdot \left[\beta_0 + \sum_{j=1}^{\infty} \beta_j \cdot \left(\sum_{k=0}^{m_O} w_{ik} x_k \right)^{2j-1} \right] \\
&\approx \alpha_0 + \sum_{i=1}^{m_I} \alpha_i \cdot \left[\frac{1}{2} + \sum_{j=1}^6 \beta_j \cdot \left(\sum_{k=0}^{m_O} w_{ik} x_k \right)^{2j-1} \right]
\end{aligned}$$

Notar que el número de términos en la última expresión en general es muy grande. Sin embargo los términos que corresponden a las neuronas escondidas pueden ser reducidos a términos de grado 0, 1, 3, 11 y en consecuencia aquellos en la neurona de salida consistirán en los que resulten de las combinaciones de potencias de grado $(0,1,3,\dots,11)^1$, $(0,1,3,\dots,11)^3,\dots,(0,1,3,\dots,11)^{11}$. Un análisis simple muestra que sólo 20 de las posibles combinaciones de potencias son posibles. Sólo se presenta el resultado del análisis con dichas combinaciones:

| | | | |
|---|----|----|-----|
| 1 | 11 | 33 | 63 |
| 3 | 15 | 35 | 77 |
| 5 | 21 | 45 | 81 |
| 7 | 25 | 49 | 99 |
| 9 | 27 | 55 | 121 |

Tabla. Combinaciones posibles de potencias impares de la expansión de la función logística

Si hacemos ponemos en una lista L las potencias anteriores, entonces el polinomio de la neurona de salida será de la forma $P_o = k + \sum_{i=1}^{20} T(i)$ en donde $T(i)$ = términos de grado $L(i)$, i -ésimo elemento de la lista.

Todo lo anterior nos ha permitido avanzar, pero aún queda el problema de cómo encontrar los coeficientes asociados a cada término.

Una manera de resolver este problema, dice *Kuri-Morales* [2], es definir a priori un número de monomios, M y posteriormente elegir cuáles de los p posibles serán los elegidos. Entonces tenemos $C(p, M)$ posibles combinaciones de monomios e incluso para valores pequeños de p una búsqueda exhaustiva no tendría sentido. Entonces abordamos el problema como uno de optimización que resolveremos con algoritmos genéticos, en específico el EGA (eclectic genetic algorithm) por tener un excelente desempeño.

El planteamiento general se expone en el siguiente apartado.

Uso de algoritmo genético para el ajuste del polinomio

Como nos dice otra vez *Kuri-Morales* [2], un cromosoma puede ser visto como un string binario de tamaño p . Cada bit en el representa un monomio ordenado como la sucesión de potencias consecutivas de las variables. Si el bit es “1” significa que el monomio correspondiente se retiene y si es “0” se descarta. Se tiene que asegura que el número de “1’s” es igual a M .

Asumir, por ejemplo, que $y = f(v_1, v_2, v_3)$ y que $d_1=1, d_2=d_3=2$. En este caso las potencias asignadas a las 18 posiciones del genoma son 000, 001, 002, 010, 011, 012, 020, 021, 022, 100, 101, 102, 110, 111, 112, 120, 121, 122. Por ejemplo, si $M=6$, el cromosoma

11000010101000001 correspondería al polinomio

$P(v_1, v_2, v_3) = c_{000} + c_{001}v_3 + c_{020}v_2^2 + c_{022}v_2^2v_3^2 + c_{101}v_1v_3 + c_{122}v_1v_2^2v_3^2$. La población del EGA consiste en un conjunto de strings binarios de longitud p en donde sólo hay M 1's. Esto es, para cada genoma los monomios (correspondientes al 1) son determinados por el EGA. Después usamos el Algoritmo de Ascenso (que no discutiremos en este texto) para obtener el conjunto

de los M coeficientes que minimizan $\epsilon_{\text{MAX}} = \max(|f_i - y_i|) \forall i$. A continuación para este conjunto de coeficientes se calcula el ϵ_{RMS} . Esta es justo la función de fitness para el EGA. Los individuos se seleccionan, se cruzan y mutan para cierto número de generaciones. Al final nos quedamos con los individuos que minimizan el ϵ_{MAX} (del algoritmo de ascenso).

Entrenamiento del modelo

El modelo con las condiciones presentadas en la sección anterior se entrenó a los datos temporales usando un programa del profesor llamado GMP4 (Genetic Multivariate Polynomials). Notar que a pesar de que nuestros datos son temporales esto realmente no importa al ajustar el modelo. El algoritmo genético simplemente ve al problema como un problema de optimización. Por eso estos algoritmos resuelven problemas muy variados y son ampliamente usados.

Lo primero que hay que hacer es seleccionar los conjuntos de entrada, de entrenamiento y de prueba, así como los archivos para obtener los resultados del ajuste. La siguiente ventana muestra este proceso:



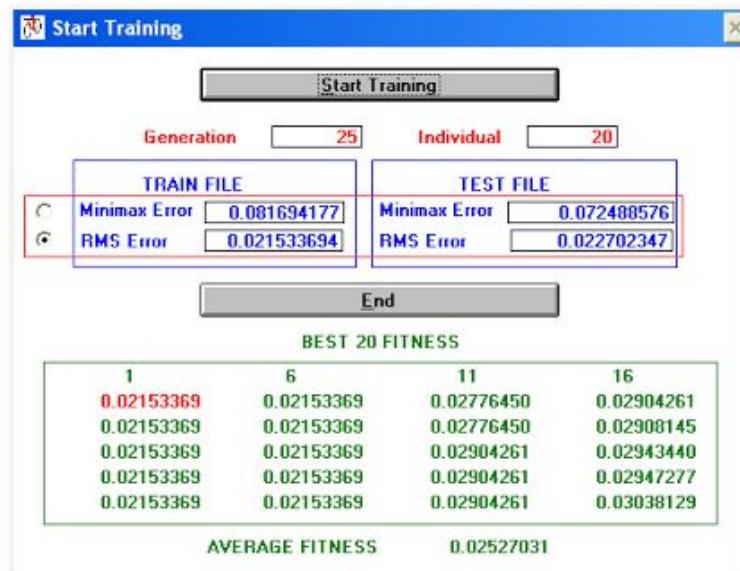
Posteriormente hay que definir los parámetros del polinomio genético a entrenar. Estos parámetros se muestran en la siguiente ventana e indican que:

- el error que usaremos es el ECM (Error cuadrático medio)
- nuestro polinomio tiene a lo más 8 términos
- el grado de los monomios es de a lo más 121
- la probabilidad de cruzamiento es de 90%
- la probabilidad de mutación es de 0.5% (debe ser baja)
- la población inicial es de 20 individuos
- el algoritmo se corre hasta tener 25 generaciones

| Name | Value |
|---------------------------|----------|
| Maximum allowed degree | 11 |
| Minimization Option | RMS |
| Highest Monomial's Degree | 121 |
| Ind. Variables | 6 |
| Terms | 8 |
| Spread Data | YES |
| QuasiMinMax | YES |
| Quasi % | 0.0500 |
| Regularize | YES |
| Regularization Factor | 0.000001 |
| Pc | 0.9000 |
| Pm | 0.0050 |
| Individuals | 20 |
| Generations | 25 |

Una vez que hemos definido los parámetros del algoritmo, este se corre y los resultados obtenidos son los siguientes:

- El error de entrenamiento es de 0.021 aprox.
- El error de prueba es de 0.022 aprox.
- El fitness promedio de todas las generaciones es de 0.025 aprox.



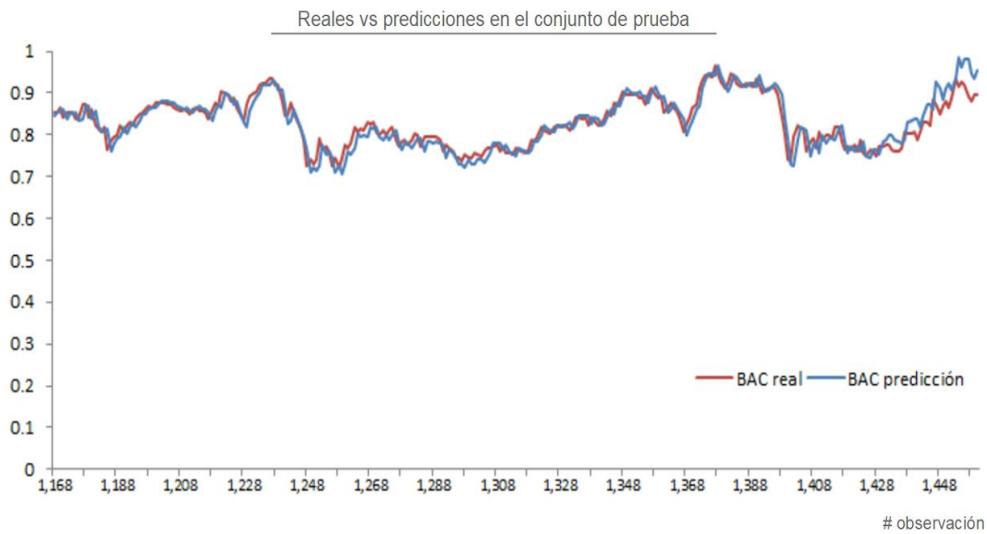
El modelo en el conjunto de prueba tiene los siguientes coeficientes:

| | A | B | C |
|---|--------|--------------------|-------------|
| 1 | EpsiTh | 0.08169418 | |
| 2 | 1 | C00,00,00,00,01,00 | 0.00825854 |
| 3 | 1 | C00,01,00,00,00,00 | -0.05572851 |
| 4 | 1 | C00,00,00,00,00,01 | 1.01719357 |
| 5 | 5 | C03,00,00,00,00,02 | 0.22364111 |
| 6 | 5 | C01,00,00,02,00,02 | -0.2586646 |
| 7 | 5 | C00,04,00,00,01,00 | 0.14295535 |

A continuación se muestra la predicción del modelo para las primeras observaciones del conjunto de prueba usando los coeficientes generados por el modelo:

| 1 | GE_lag_1 | PBR_lag_1 | PFE_lag_1 | S_lag_1 | SUNE_lag_1 | BAC_lag_1 | BAC real | BAC predicción |
|----|------------|------------|------------|------------|------------|------------|------------|----------------|
| 2 | 0.72168993 | 0.33446858 | 0.7328205 | 0.54901961 | 0.61654133 | 0.84780175 | 0.85344424 | 0.847462108 |
| 3 | 0.71106556 | 0.31730978 | 0.72403323 | 0.53979239 | 0.58711995 | 0.85344424 | 0.85485476 | 0.853810163 |
| 4 | 0.7080301 | 0.31244383 | 0.71884065 | 0.52479815 | 0.58058184 | 0.85485476 | 0.86402386 | 0.855577574 |
| 5 | 0.70347679 | 0.31526097 | 0.72922569 | 0.52133795 | 0.61261852 | 0.86402386 | 0.8407486 | 0.862588517 |
| 6 | 0.68425186 | 0.30322417 | 0.7160447 | 0.49250288 | 0.58450472 | 0.8407486 | 0.85344424 | 0.838762319 |
| 7 | 0.68829916 | 0.32627328 | 0.70565969 | 0.49596309 | 0.58352398 | 0.85344424 | 0.85203358 | 0.852403312 |
| 8 | 0.67767494 | 0.28119944 | 0.70765686 | 0.48788927 | 0.59692705 | 0.85203358 | 0.85485476 | 0.84832872 |
| 9 | 0.68779337 | 0.26813827 | 0.69966837 | 0.48673587 | 0.56685186 | 0.85485476 | 0.83863265 | 0.853492342 |
| 10 | 0.66452102 | 0.24534523 | 0.68289256 | 0.47635525 | 0.54723767 | 0.83863265 | 0.84286448 | 0.834447977 |
| 11 | 0.66249742 | 0.24765015 | 0.67929775 | 0.47635525 | 0.54756458 | 0.84286448 | 0.8717823 | 0.838468326 |
| 12 | 0.67666309 | 0.26250403 | 0.68568851 | 0.47635525 | 0.55835242 | 0.8717823 | 0.8717823 | 0.869259001 |
| 13 | 0.66755652 | 0.30937061 | 0.68369143 | 0.44752018 | 0.54494928 | 0.8717823 | 0.84286448 | 0.870346338 |
| 14 | 0.64681383 | 0.32627328 | 0.66851335 | 0.45213379 | 0.52795033 | 0.84286448 | 0.85979203 | 0.839960206 |
| 15 | 0.66907433 | 0.32755381 | 0.69207927 | 0.455594 | 0.54985291 | 0.85979203 | 0.82241054 | 0.858716613 |
| 16 | 0.64529613 | 0.33421245 | 0.67650176 | 0.44521338 | 0.50375941 | 0.82241054 | 0.8146521 | 0.820667492 |
| 17 | 0.61949429 | 0.30476078 | 0.6820937 | 0.42445213 | 0.47270348 | 0.8146521 | 0.80900961 | 0.810125776 |
| 18 | 0.60330491 | 0.34701753 | 0.65573171 | 0.43598616 | 0.41712977 | 0.80900961 | 0.81747334 | 0.806074246 |
| 19 | 0.61089367 | 0.3426638 | 0.65533228 | 0.45213379 | 0.43347498 | 0.81747334 | 0.76386975 | 0.814478805 |

Y finalmente podemos ver la serie temporal del real vs predecido por el polinomio en el conjunto de prueba. Podemos ver que para las primeras observaciones el modelo predice bastante bien, sin embargo, para observaciones finales se puede observar que la predicción ya no es tan buena. Si bien lo anterior puede ser preocupante, quizás no lo es tanto, pues para las primeras observaciones el modelo lo hace muy bien y el modelo puede ser re-entrenado fácil y periódicamente.



Comparación de ambos modelos

- En la red neuronal el error de prueba fue de 0.014
- En el polinomio el error de prueba fue de 0.022
- Por lo tanto, en cuanto a error, la red neuronal es mejor.
- Sin embargo, el algoritmo genético puede ajustarse con más generaciones (en este caso solo usamos 25) de modo que la aproximación mejore.
- Los dos corrieron prácticamente en el mismo tiempo.
- En cuanto a los parámetros de la arquitectura, consideramos que en ambos modelos son igualmente “difíciles” de definir, pues mientras en la red neuronal debes de indicar el número de inputs, capas ocultas y neuronas en las capas, en el polinomio hay que definir los parámetros del algoritmo genético (cuántas generaciones, individuos, probabilidad de cruce, de mutación, etc).
- Por lo tanto, consideramos ambos modelos igualmente buenos.
- Pero un argumento a favor del polinomio es que es una forma funcional cerrada y no un “modelo de caja negra” como la red neuronal. El algoritmo genético es un poco complicado de explicar a audiencia no especializada pero creemos que la idea de un modelo polinomial es relativamente fácil de explicar.
- Y un argumento a favor de la red neuronal es el hecho de que en el contexto de precios de acciones, en donde un cambio en el precio tiene una repercusión millonaria o

billonaria, quizá es preferible tener una mayor precisión aunque usemos modelos de “caja negra”.

Conclusiones

Es claro que el tratamiento de series de tiempo es complejo en su naturaleza. No sólo hay que asegurarse de que las variables que se están utilizando para su modelado cumplen con las características deseables que pudieran buscarse en cualquier ejercicio de predicción, si no que también, es necesario identificar las posibles correlaciones temporales que pudieran existir entre estas. Más aún, es necesario, por medio de un proceso iterativo, incluir diversos niveles de rezagos y verificar tanto por medio de estadística descriptiva cómo por medio de técnicas de regularización y pruebas de ajuste cuales son las variables más útiles para el análisis.

En este sentido, y por consecuencia del mismo, gran parte del documento se dedicó a la elección y procesamiento de variables. Diversos algoritmos se desarrollaron para la construcción de variables rezagadas y se llevaron a cabo distintas pruebas para determinar los rezagos más correlacionados con la variable respuesta. Además de esto, se exploraron diversos métodos para reducir la dimensionalidad de la base de datos de manera algorítmica. En primer lugar se utilizó la técnica no supervisada de PCA y se encontró, corroborando los resultados del análisis exploratorio, que gran parte de la variabilidad de los datos se concentraba en un par de variables. Para futuras indagaciones, con tal de compensar las deficiencias inherentes de PCA, podrían utilizarse mínimos cuadrados parciales o LPS por sus siglas en inglés, para reducir la dimensionalidad de la base tomando en consideración la correlación con respecto a la variable respuesta. Además de esto, se corrieron redes neuronales con tal de eliminar las variables que no ayudarán al modelo a predecir. Tanto la red neuronal como los modelos polinomiales probaron tener alta precisión y al mismo tiempo desempeño satisfactorio.

Referencias

- [1] A. Kuri-Morales, P. Kuri-Morales. *Forecasting epidemiological behavior of respiratory diseases in Mexico city using neural networks*, ITAM-Secretaría de Salud, México.
- [2] A. Kuri-Morales, A. Cartas-Ayala. *Polynomial Multivariate Approximation with Genetic Algorithms*, ITAM, México.
- [3] L. Maciel, R Ballini. Design a Neural Network for time series financial forecasting: accuracy and robustness analysis, Universidad de Estado de Campinas, Brasil.
- [4] Einstein, A. 1914 Méthode pour la détermination des valeurs statistiques d'observations concernant des grandeurs soumises à des fluctuations irrégulières. *Arch. Sci. Phys. et Naturelles Ser. 4* 37 254-255.
- [5] Hooker R H 1901 Correlation of the marriage-rate with trade *J Royal Statist Soc* 64 696-703.
- [6] Wiener N 1949 *Time Series* MIT Press, Cambridge.
- [7] Mukherjee, Sayan, Edgar Osuna, and Federico Girosi. "Nonlinear prediction of chaotic time series using support vector machines." *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*. IEEE, 1997.
- [8] Härdle, Wolfgang, Helmut Lütkepohl, and Rong Chen. "A review of nonparametric time series analysis." *International Statistical Review* 65.1 (1997): 49-72.
- [9] Cybenko, George. "Approximation by superpositions of a sigmoidal function." *Mathematics of control, signals and systems* 2.4 (1989): 303-314.
- [10] Coglan, Avril. "A little book of R for Multivariate Analysis" [en línea]. U.K.: Wellcome Trust Sanger Institute, Cambridge, August 2014 [fecha de consulta: 21 noviembre 2015]. Capítulo II. Disponible en <<http://little-book-of-r-for-multivariate-analysis.readthedocs.org/en/latest/src/multivariateanalysis.html>>
- [11] Wikipedia. "Principal Component Analysis" [en línea]. Actualización: 17 Noviembre 2015 [fecha de consulta: 19 noviembre 2015]. Introducción. Disponible en <https://en.wikipedia.org/wiki/Principal_component_analysis>

- [12] Hall, Mark A. and Smith, Lloyd A. "Feature Subset Selection: A Correlation Based Filter Approach" [en línea]. New Zealand: Department of Computer Science, University of Waikato, Hamilton. [fecha de consulta: 22 noviembre 2015]. Introducción, Capítulo III. Disponible en <<http://www.cs.waikato.ac.nz/ml/publications/1997/Hall-LSmith97.pdf>>
- [13] Perez-Riverol, Yasset. "Introduction to feature selection for bioinformaticians using R, correlation matrix filters, PCA & backward selection" [en línea]. U.K: Polytechnic University Havana. October 2013 [fecha de consulta: 22 noviembre 2015]. Introducción. Disponible en <<http://computationalproteomic.blogspot.mx/2013/10/introduction-to-feature-selection-for.html>>
- [14] Fahlman, S.E: An empirical study of learning speed in back propagation networks. Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufman 1988.
- [15] Reed, R: Pruning Algorithms A Survey. IEEE Trans. On Neural Networks, Volume 4, Number 5, 1993, pp. 707-740.