Maestría en Ciencia de Datos
Departamento de Matemática y Ciencias

Análisis de Series de Tiempo de Funciones de Densidad

Con aplicación al sector aeroespacial

Nicolás Lupi **2023**

Directora: Marcela Svarc

Universidad de
SanAndrés

Resumen

El objetivo del trabajo es comparar distintos métodos para proyectar la evolución de una función de densidad en el tiempo. En concreto, dado un conjunto de observaciones generadas por un proceso aleatorio a lo largo del tiempo, nos interesa caracterizar la evolución del mismo para poder predecirlo a futuro. La motivación es poder generar muestras de una nueva distribución condicional al paso del tiempo, si bien podría condicionarse a otra variable según el problema. El disparador de este análisis es un estudio que se quiere realizar sobre el cambio a lo largo de los años de las características que tienen los lanzamientos de satélites a órbita, en particular sobre la masa de los mismos. A partir de los datos, para cada año se puede obtener un estimador de la densidad de la masa de los satélites. En base a estas estimaciones se busca predecir las densidades para los próximos años.

Se proponen tres métodos alternativos. El primer enfoque es paramétrico y asume que la secuencia de distribuciones proviene de una familia paramétrica y que las observaciones futuras también pertenecerán a la misma familia. Luego, el problema se reduce en modelar mediante series de tiempo los parámetros de dichas distribuciones. El segundo enfoque es no paramétrico y se basa en el Análisis de Componentes Principales Funcionales (FPCA en inglés): considerando las densidades como una secuencia de datos funcionales, los mismos serán representados en una base conveniente de funciones ponderadas por escalares que variarán a lo largo del tiempo. Con estos escalares se construirá una serie de tiempo para proyectar a futuro. Por último, el método de Redes Adversarias Generativas Condicionales (cGAN por sus siglas en inglés) buscará entrenar un modelo para generar muestras nuevas similares a las reales teniendo en cuenta el período para el cual serán generadas. Al igual que el segundo método, este no hace ningún supuesto sobre las distribuciones, pero a diferencia del anterior no requiere construir ni proyectar ninguna serie de tiempo.

Este trabajo presenta la siguiente estructura. En el Capítulo 1 se introduce la motivación del problema y en el Capítulo 2 se describen las tres propuestas metodológicas. El Capítulo 3 está abocado a analizar sus desempeño en conjuntos de datos simulados. En el cuarto capítulo se analizan los datos correspondientes a los pesos de los satélites. Finalmente, se presentan las conclusiones del trabajo.

Agradecimientos

Agradezco a Marcela Svarc por su paciencia, sugerencias y ayuda.

Dedicado a Flor, Fue en Defensa Propia y amigos.

"Si esto va de a tres, a cada paso haremos cien . . . "

Índice general

CAPÍTULO 1

Introducción

Muchas veces nos encontramos con datos sobre los cuales nos interesa poder predecir su comportamiento en el futuro. Un caso usual de este problema consiste en tener muestras de algún proceso estocástico registradas en distintos momentos del tiempo, las cuales agrupamos en ventanas resumiendo la información obteniendo los estadísticos usuales como la media, mediana, desvío, percentiles, máximo, mínimo, entre otros.

Por ejemplo, podemos tener datos meteorológicos como la temperatura de una ciudad, dato que podríamos disponer con frecuencia horaria. Puede ser de nuestro interés querer predecir la evolución en el tiempo de esta variable, para lo cual una opción sería resumir las observaciones en frecuencia diaria, tomando por ejemplo el promedio diario, y construyendo una serie de tiempo con un dato por día para luego proyectar a futuro.

Un ejemplo de otra índole puede ser datos sobre las órbitas a las que se lanzaron satélites a lo largo del tiempo. De estos satélites podemos estar interesados en sus masas, para decir si con el pasar de los años se volverán más livianos o más pesados; o en los parámetros que determinan la órbita como la altitud, inclinación, entre otros, para poder determinar si existe algún patrón en el tiempo. Una vez más, un enfoque podría ser resumir todos estos datos en el "lanzamiento promedio" de cada año y luego construir una serie de tiempo en base a este promedio.

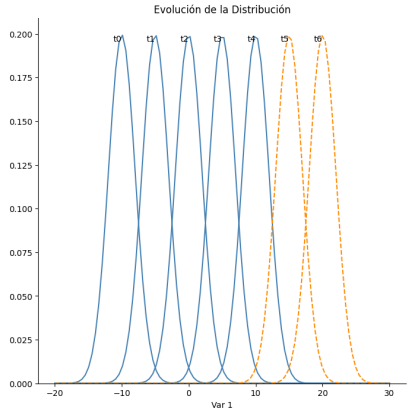
A menudo, para el análisis de este tipo de datos basta con construir una serie temporal de los estadísticos de interés y modelar la serie en forma tradicional. Sin embargo, este enfoque puede ser insuficiente, por ejemplo si nuestros datos presentan más de una moda con comportamientos distintos que se ven ocultadas al tomar los estadísticos clásicos. En el caso de los satélites, podríamos resumir la información en la masa promedio, y ver que la misma no varió entre un año y otro, cuando en realidad lo que estaba sucediendo era que teníamos dos modas separándose una de la otra, las cuales podrían estar registrando algún patrón en el tiempo que nos estaríamos perdiendo. Esta idea se puede comprender mejor observando la Figura 1.1. De aquí surge la necesidad de adoptar otro enfoque que nos permita conocer la ley de distribución de nuestros datos y cómo la misma varía en el tiempo.

Este trabajo consiste en aplicar, al análisis temporal de datos, métodos que permitan extraer información útil de las observaciones como un todo. De esta manera, podremos no solamente decir cómo se comportará la temperatura o el lanzamiento promedio, sino que podremos sacar conclusiones de la distribución que generó esos datos y cómo fue variando en el tiempo. En particular se hará

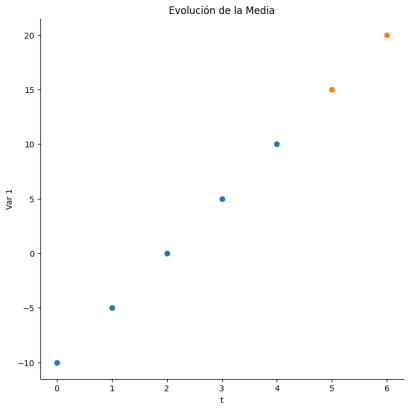
énfasis en métodos que permitan poder generar nuevas observaciones para períodos futuros.

Nuestro problema podría consistir en realizar simulaciones para un sistema complejo que dependa de las condiciones climáticas, como el caso de alguna entidad interesada en saber si la red de desagüe de una ciudad podría dar abasto frente a las lluvias del próximo año. Para nada serviría saber que el nivel de precipitación promedio del período siguiente será de tantos milímetros, aún si nuestra red hipotética pudiese tolerarlo, si pasamos por alto que niveles no tolerables son altamente probables dadas las lluvias históricas.

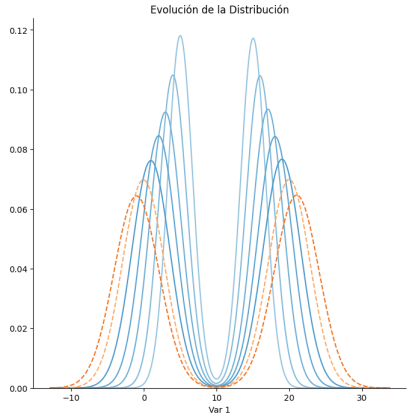
También podríamos ser una entidad encargada de operar o lanzar satélites y necesitamos tener una idea respecto a qué características tendrán estos en el futuro. De nuevo, trabajando únicamente con estadísticos de resumen estaríamos perdiendo información valiosa, lo que nos podría llevar a tomar decisiones desacertadas.



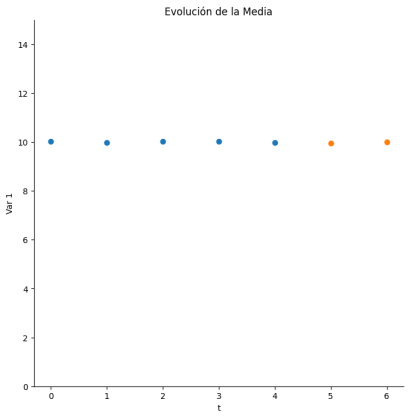
(a) Desplazamiento de la distribución



(b) Aumento de la media



(c) Separación de modas



(d) Media constante

Figura 1.1: En la primera fila, las distribuciones se desplazan en el tiempo dando lugar al aumento en la media; en este caso no perderíamos información al predecir únicamente la media. En la segunda fila, tenemos distribuciones bimodales cuyas modas se alejan una de la otra con el paso del tiempo; en este caso, la media no es un buen reflejo del comportamiento de las distribuciones.

CAPÍTULO 2

Marco Teórico

2.1. Abordaje del Problema

Partimos de un conjunto de observaciones de las cuales $X_{i_t,t}$ representa a la observación i en el período t. Notar que en nuestro caso la variable t representa el período de tiempo en el que se originó esa observación y varía del período 1 a T, pero podría adaptarse a cualquier problema en el que queramos estimar una función de densidad condicionando sobre alguna variable. Notar también el subíndice $i_t = 1, \ldots, n_t$, para remarcar que las muestras en distintos períodos no guardan ninguna relación entre sí (no son datos de panel), e incluso el tamaño de las muestras no necesariamente es siempre el mismo en los distintos momentos.

Suponemos que estas observaciones fueron generadas por alguna distribución con función de densidad f_t que va variando en el tiempo (o según lo que represente t). Esta función es desconocida para nosotros, por lo que la estimamos para cada t en base a nuestras observaciones, obteniendo $\hat{f}_t$. Luego el problema consiste, en base a $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_T$, en poder predecir $\hat{f}_{T+h}$ para el h que se desee.

Para esto vamos a presentar tres métodos alternativos y probarlos en datos simulados y en datos reales de lanzamientos de satélites. En primer lugar partiremos del caso en el cual supondremos que las f_t pertenecen a alguna familia de distribuciones paramétrica conocida, ya sea por algún conocimiento sobre el problema o porque los datos lo sugieren. De esta forma podremos estimar para cada período t los parámetros de la distribución y predecir los mismos mediante técnicas estándares de series temporales.

La segunda alternativa consiste en utilizar elementos del campo de análisis de datos funcionales (FDA). Sin la necesidad de suponer que las funciones provienen de alguna familia en particular estimaremos las $\hat{f}_t$ para cada t ya sea de forma paramétrica o no paramétrica. Una vez estimadas estas funciones, les aplicaremos componentes principales funcionales, descomponiéndolas así en una serie de funciones subyacentes invariantes en el tiempo, que ponderadas con distintos pesos darán lugar a las funciones originales. Estos pesos son los que varían en el tiempo, y con ellos construiremos la serie de tiempo para proyectar las funciones a futuro.

Por último, consideraremos una alternativa utilizando redes neuronales, entrenando un modelo de redes adversarias generativas condicionales. Un modelo intentará generar nuevas observaciones que se parezcan lo más posible a un dato real para un momento dado, y el otro intentará distinguir dicha observación de las muestras originales, teniendo en cuenta el momento del tiempo al cual se está condicionando. A diferencia de los anteriores, este método no nos permitirá

obtener explícitamente la función de densidad ni para los períodos $1, \ldots, T$ ni para los posteriores. El output será un modelo generativo capaz de generar muestras para períodos futuros, sobre las cuales ajustaremos las densidades $\hat{f}_{T+h}$ como último paso.

2.2. Metodología

Método Paramétrico

Una idea simple consiste en suponer que las observaciones provienen de una distribución paramétrica $f(\cdot, \theta_t), \theta_t \in \Theta$. No sólo eso, sino que en cada período la distribución pertenece a la misma familia y lo que varía en el tiempo son los valores de los parámetros que la caracterizan.

Como en la práctica los parámetros θ_t no son observados, el primer paso consiste en ajustar las distribuciones para cada período en base a los datos disponibles, estimando los parámetros mediante algún método como máxima verosimilitud y dando lugar a $f(\cdot, \hat{\theta}_t)$.

Con las estimaciones se construye la serie $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_T$ con la que luego obtendremos $\hat{\theta}_{T+h}$ para el h que se desee. Proyectando los parámetros a futuro, y manteniendo el supuesto de que las funciones seguirán perteneciendo a la misma familia, construiremos la función $f(\cdot, \hat{\theta}_{T+h})$.

Si bien el método es simple, ya pueden verse algunas de sus desventajas. Por ejemplo, requiere tener algún conocimiento o hacer supuestos sobre la familia de la que provienen las distribuciones. Además, algunos parámetros requieren cierto cuidado a la hora de proyectarlos a futuro: si proyectamos un desvío deberemos imponer la restricción de que sea siempre positivo; o si proyectamos una matriz de covarianza la misma deberá ser siempre semidefinida positiva. El método sería adecuado sólamente si los datos se ajustan en forma correcta a una familia de distribuciones. Si no se encuentra un ajuste paramétrico adecuado, o si hacerlo requiere la estimación de un número cada vez mayor de parámetros, esta propuesta deja de resultar conveniente y pasaríamos a considerar alguna alternativa no paramétrica.

Método No Paramétrico - PCA Funcional

En segundo lugar consideramos la alternativa propuesta por Sen y Ma [11], donde en lugar de proyectar los parámetros estimados de una función, estiman y proyectan la función como un todo, recurriendo al campo de estadística de datos funcionales (FDA). La idea, al igual que antes, consiste en partir de la serie de funciones de densidad para cada período $f_1, f_2, \ldots, f_T$ para poder proyectar f_{T+h}. La diferencia es que, ahora, el método parte de estimar las funciones f_t sin la necesidad de suponer la familia de la cual provienen. Siguiendo a Horta y Ziegelmann [5], supondremos que la serie $f_1, f_2, \ldots$ es estacionaria y tomaremos a las f_t como elementos del espacio de Hilbert $L^2(I)$ definidas en un intervalo compacto I y dotadas del producto interno $\langle f, g \rangle := \int_I f(x)g(x)dx$.

Al ser f_t no observable, en la práctica debemos estimar primero $\hat{f}_t$ en base a las observaciones $\{x_{1,t}, \ldots, x_{n_t,t}\}$. En los casos de aplicación citados, los autores proponen estimar las densidades con kernels, pero a priori podría utilizarse cualquier método (siempre y cuando se mantenga cierta calidad en el ajuste).

En nuestro caso tomamos este camino tanto para las simulaciones como para el caso de aplicación, estimando las densidades con kernel gaussiano:

$$\hat{f}_t(s) = \frac{1}{n_t h_t} \sum_{i=1}^{n_t} K\left(\frac{s - x_{i,t}}{h_t}\right),$$

donde $K(s) = (\sqrt{2\pi})^{-1} \exp(-s^2/2)$ y h_t es el ancho de banda para el cual tomamos la regla de Silverman [12] $h_t = 1{,}06\,\hat{\sigma}_t\, n_t^{-1/5}$ donde $\hat{\sigma}_t$ es el desvío estándar calculado para las observaciones del período t.

El siguiente paso consiste en aplicar Análisis de Componentes Principales Funcionales (FPCA) a las T funciones de densidad ajustadas en el paso anterior, pero para explicar mejor este punto primero haremos una analogía con el Análisis de Componentes Principales (PCA) tradicional, siguiendo a Ramsay y Silverman [10] y a Nicol [9]. Dado el vector aleatorio $X = (X_1, \ldots, X_p)$[1], PCA consiste en hallar un subespacio lineal de menor dimensión tal que la varianza de los datos proyectados a este subespacio sea máxima. En primer lugar, se busca el vector unitario ξ_1 de manera que la proyección de X a este vector tenga la mayor varianza posible:

$$\max_{\xi_1} \operatorname{Var}\left(\xi_1^{'} X\right) = \xi_1^{'} S \xi_1 \quad \text{s.a.} \quad \xi_1^{'} \xi_1 = 1,$$

donde S es la matriz de covarianza de X. La solución al problema anterior es el autovector de S asociado al mayor autovalor. El siguiente paso es hallar el vector unitario ξ_2 buscando maximizar la varianza de los datos proyectados al mismo, esta vez imponiendo además que ξ_2 sea ortogonal a ξ_1. El proceso puede repetirse entonces quedándose con los autovectores de S asociados a los autovalores más grandes, y en cada paso la variación marginal que se agregue irá decayendo.

En términos muestrales, partiendo de una muestra aleatoria x_i con $i = 1, \ldots, n$, el primer paso de PCA consiste en hallar el vector ξ_1 de manera que la expresión $y_{i,1} = \xi_1^{'} x_i$ - la proyección en la dirección del primer componente principal - tenga la mayor varianza posible $N^{-1} \sum_i y_{i,1}^2$, sujeto a que el vector ξ_1 tenga norma unitaria. Se puede repetir el paso anterior hallando nuevos ξ_j, imponiendo en cada paso que ξ_j sea ortogonal a ξ_k con $k = 1, \ldots, j-1$. La matriz S será reemplazada por su variante muestral, Σ_n de dimension p x p donde el elemento $a_{i,j} = \frac{1}{n} \sum_k^n x_k^i x_k^j$, y la solución consistirá en quedarse con los autovectores de esta última matriz, ordenados según sus autovalores.

En el caso de FPCA, en lugar de partir de un vector aleatorio de dimensión finita, partimos de una variable funcional $f(x)$; en lugar de matrices tendremos operadores lineales; en lugar de sumatorias tendremos integrales; y en lugar del producto interno tradicional utilizaremos el producto interno del espacio L^2. En este contexto usamos las siguientes definiciones:

$$\mu(x) = \mathbb{E}\left[f(x)\right],$$

$$\operatorname{Cov}_f(x, y) = \sigma(x, y) = \mathbb{E}\left[f(x) f(y)\right] - \mathbb{E}\left[f(x)\right] \mathbb{E}\left[f(y)\right],$$

$$\operatorname{Var}_f(x) = \sigma^2(x) = \mathbb{E}\left[f(x)^2\right] - \mathbb{E}\left[f(x)\right]^2$$

[1] Por simplicidad, para la explicación suponemos que $\mathbb{E}[X] = 0$, pero podría no serlo.

Definimos también al siguiente operador de covarianza:

$$\Gamma : L^2 \longrightarrow L^2, \quad \Gamma(f) = \int_I \sigma(x,y) f(x) dx$$

Dado nuestro elemento aleatorio $f(x)$, podemos plantear el problema de PCA en términos funcionales, donde buscaremos proyectar dicha función aleatoria a una base de funciones ortogonales ξ_i de manera que la varianza de $\langle \xi_i, f \rangle$ sea máxima:

$$\max_{\xi_i \in L^2} \text{Var}\left(\langle \xi_i, f \rangle\right) \quad \text{s.a.} \quad \langle \xi_i, \xi_k \rangle = \delta_{ik},\ k \leq i,\ i = 1, 2, \ldots$$

Continuando la analogía con PCA tradicional, la solución al problema anterior se dará cuando $\xi_i(x)$ sean las autofunciones (en lugar de autovectores) del operador de covarianza $\Gamma(f)$. Estas ξ_i son los componentes principales funcionales, y las nuevas variables $y_i = \langle f - \mu, \xi_i \rangle$ son las proyecciones de f en la dirección de ξ_i, o los *scores* de los componentes principales. Acto seguido, podemos reescribir a $f(x)$ según la expansión de Karhunen-Loève:

$$f(x) = \mu(x) + \sum_{k=1}^{\infty} y_k \xi_k(x)$$

En la práctica, en lugar de los puntos x_i nuestra muestra consistirá de una serie de n funciones f_i que provendrán del mismo proceso aleatorio. En términos muestrales, ahora buscaremos maximizar $\frac{1}{n}\sum_{j=1}^{n} \langle f_j, \xi_k \rangle^2$ imponiendo que las ξ_k sean ortogonales entre sí. La solución se dará para las autofunciones de la versión muestral del operador de covarianza $\hat{\Gamma}_n(f) = \frac{1}{n}\sum_{j=1}^{n} \langle f_j, f \rangle f_j$.

Dada nuestra serie de funciones $f_1, f_2, \ldots, f_T$, definimos entonces al *score* $y_{t,k} = \langle f_t - \hat{\mu}, \xi_k \rangle$. Finalmente, nuestra versión empírica de la expansión de Karhunen-Loève queda como:

$$f_t(x) = \hat{\mu}(x) + \sum_{k=1}^{T} y_{t,k} \hat{\xi}_k(x)$$

Al igual que en PCA tradicional, la varianza explicada decae con cada ξ_k que se agregue, por lo que se espera que un número pequeño $K < T$ de componentes debería bastar para representar suficientemente bien a las funciones. De alguna forma lo que estamos haciendo es proyectar los datos a una dimensión menor. Ahora podemos escribir nuestras funciones de densidad como una combinación entre una serie de K funciones (los elementos de nuestra nueva base), ponderadas en cada caso por sus correspondientes pesos, los $y_{t,k}$. Lo interesante de esto es que, una vez halladas las funciones $\hat{\xi}_k$, podemos analizar nuestras densidades en el espacio de los escalares $y_{t,k}$:

$$\hat{f}_t(x) \approx \hat{\mu}(x) + \sum_{k=1}^{K} y_{t,k} \hat{\xi}_k(x)$$

Volviendo al contexto de una serie de tiempo de T densidades, lo que hacemos es aplicar FPCA para representarlas como una combinación de K funciones, que en cada período deben ser ponderadas por un vector de pesos y_t para reconstruir la función original. Lo importante es que, dado este análisis, lo único

que varía en el tiempo es el vector y_t. Por lo tanto, logramos llevar el problema de una serie de tiempo de funciones, a uno de serie de tiempo de escalares que puede ser abordado con herramientas tradicionales. Así, podremos proyectar a futuro al vector y_{T+h}, con el cual ponderaremos a las K funciones de nuestra base, para dar con lo que sería la densidad $\hat{f}_{T+h}$.

En otras palabras, y siguiendo la analogía con PCA tradicional, este método sería análogo a tener datos en $\mathbb{R}^n$ generados cada uno en momentos del tiempo distintos. El método consistiría en reducir la dimensión a $\mathbb{R}^K$, y luego hacer el análisis de serie de tiempo con las proyecciones.

Los pasos descritos arriba alcanzarían para una serie de tiempo de funciones. Sin embargo, en nuestro caso lidiamos con funciones de densidad, las cuales traen restricciones de no negatividad e integral unitaria. Para imponer la primera restricción, en lugar de aplicar FPCA sobre las densidades lo hacemos sobre el logaritmo de las mismas. Luego, cuando queremos reconstruir f_t tomamos la exponenciación de la expresión que tengamos [11]. Para imponer la segunda restricción, dividimos la función resultante por su integral.

GANs Condicionales

Por último, consideraremos como alternativa utilizar redes adversarias generativas condicionales (cGANs). Una GAN es un procedimiento de aprendizaje no supervisado donde se busca determinar los patrones de un conjunto de datos, para luego poder generar nuevas muestras que podrían haber conformado el conjunto original de observaciones. La idea de este método será conseguir un modelo generador capaz de, en base a observaciones provenientes de distribuciones de períodos anteriores, poder generar nuevas observaciones de las distribuciones futuras.

Para ajustar una GAN se parte de un conjunto de datos, un modelo generador (G) y un modelo discriminador (D). El modelo generador tiene como objetivo generar observaciones que recuperen las características principales de los datos originales, partiendo de un vector de ruido. El modelo discriminador debe identificar cuáles datos son verdaderos y cuáles no. De esta manera, el problema típico que atacan las GANs es el que introdujeron Goodfellow et al. [3]:

$$\min_G \max_D \mathbb{E}_{x\sim p_X(x)} \left[log\, D(x)\right] + \mathbb{E}_{z\sim p_z(z)} \left[log\left(1 - D(G(z))\right)\right], \qquad (2.1)$$

donde $D(\cdot)$ es la salida del modelo discriminador (cuán verosímil es la observación en cuestión), $G(\cdot)$ es la salida del modelo generador (vector que vive en el espacio de las observaciones de la muestra) y p_X y p_z son las distribuciones poblacional de los datos, y de ruido, respectivamente. En otras palabras, tenemos un problema de *minimax* entre dos jugadores donde uno, el discriminador, quiere maximizar la función (2.1), es decir, detectar a los datos reales y a los "falsos" como tales, mientras que el otro, el generador, busca exactamente lo contrario, intentando minimizar la función (2.1) de manera que el discriminador esté tomando a los datos generados como verdaderos.

A medida que el discriminador distinga mejor entre datos verdaderos y falsos, el valor de la función objetivo se acercará a cero. En cambio, si el generador logra entrenarse para engañar al discriminador y dejarlo sin mejor alternativa que "adivinar" si una muestra es real o falsa, la función objetivo alcanzaría un valor de $-log(4)$. Esto puede verse teniendo en cuenta primero que, dejando fijo

a un generador que de lugar a la distribución sobre el dominio de los datos $p_g(x)$, el valor mínimo que podría alcanzar la función objetivo se encontraría cuando el discriminador produzca $D^*_G(x) = \frac{p_X(x)}{p_X(x)+p_g(x)}$ (ver [3]). Si el generador logra entrenarse hasta lograr un mapeo perfecto entre Z y X, entonces tendremos que $p_g = p_X$, por lo que $D^*_G = 0{,}5$. Dicho de otra manera, si el generador se entrenase a la perfección, cuando un discriminador óptimo se encontrase con una nueva observación no tendría manera de saber si es falsa o verdadera, teniendo entonces que elegir al azar con una probabilidad de 0,5 de acertar.

Entrenar una GAN consiste en gran parte en mantener una sincronización entre la capacidad que van adquiriendo el discriminador y el generador. Es importante balancear el ritmo al cual ambos modelos se entrenan para evitar equilibrios indeseados como el "Escenario Helvética" [3], donde un generador que se entrena muy por encima del discriminador rival, termina aprendiendo únicamente a colapsar el espacio Z a un único punto del dominio de X donde el discriminador no puede determinar si la observación es real o falsa. En nuestro caso de uso de las GANs que detallaremos más adelante, este problema sería equivalente a que el modelo generador aprendiera a generar únicamente la moda de nuestra distribución. Estas anomalías suelen darse con estos modelos y es importante estar atentos a ellas.

En general, las redes adversarias suelen entrenarse con el objetivo de tener un modelo generativo de observaciones que viven en espacios muy complejos, como en el caso de la generación de imágenes. En línea con lo anterior, existen las GANs condicionales, desarrolladas por Mirza y Osindero [8]. Como su nombre lo indica, el problema que buscan abordar las cGANs es el de la generación de muestras condicional a una o más variables y. El discriminador debe tener en cuenta la condición de la variable y para discernir si una muestra es verdadera o falsa según el contexto, y el generador debe considerarla para generar muestras lo más reales posibles dado ese mismo contexto. De esta manera la función objetivo pasa a ser:

$$\min_G \max_D \mathbb{E}_{x\sim p_X(x)} \left[log\, D(x|y)\right] + \mathbb{E}_{z\sim p_z(z)} \left[log\left(1 - D(G(z|y))\right)\right]$$

Por ejemplo, supongamos que queremos entrenar una GAN para generar imágenes de animales. Una GAN estándar se entrenaría con imágenes de varios tipos de animales y a la hora de generar nuevas observaciones no tendríamos ningún control sobre las imágenes resultantes. En cambio, con una cGAN podríamos indicarle al generador si queremos generar la imagen de un perro o un gato, y al discriminador que juzgue si la imagen es real o falsa teniendo en cuenta que debería ser de un perro o un gato. En otras palabras, estaríamos etiquetando al set de datos para que el generador aprenda cómo cambiar el mapeo de Z a X según el valor de estas etiquetas, y para que el discriminador aprenda de alguna manera a determinar cómo deberían verse dichas etiquetas para detectar las observaciones del generador.

En los ejemplos típicos de cGANs, la variable y es categórica (como el dígito de la base de datos MNIST[2] que queremos generar, o la estación del año para la cual queremos producir la imagen de un parque). Fabbri [2] propone además casos donde y es numérica, como el grado de polvo que debe aparecer en la

[2]Modified National Institute of Standards and Technology; http://yann.lecun.com/exdb/mnist/

imagen de una galaxia, o la edad de la persona para la cual queremos generar un retrato.

Nuestra propuesta es utilizar las cGANs no para la generación de imágenes como se hace habitualmente, sino para la generación de observaciones condicional al momento del tiempo que seleccionemos. Una vez terminado el entrenamiento, tendremos un modelo generativo al cual le pasaremos la variable y con el momento del tiempo que fijemos, para que devuelva muestras aleatorias que, de acuerdo al modelo, podrían haberse realizado en ese tiempo. Luego, suponiendo que el generador aprendió cómo cambiar el mapeo de Z a X según el cambio en t, podremos pedirle que genere muestras para $T + h$, consiguiendo así un modelo capaz de producir observaciones futuras, en base a lo aprendido de las observaciones pasadas.

Una diferencia respecto a las dos primeras propuestas es que, en este caso, no vamos a predecir la densidad sino que lo que haremos será generar nuevas muestras $x_{i,t}$ con las cuales luego podremos ajustar una distribución con los métodos tradicionales.

Otra consideración a tener en cuenta es que entrenar una GAN con la función objetivo tradicional (2.1) suele acarrear algunos problemas como el colapso sobre la moda que mencionamos antes, o la inestabilidad del modelo ante cambios sutiles en los parámetros, debido a que (2.1) suele no ser continua en los pesos del generador. Para solucionar este problema, Arjovsky et al. proponen utilizar la distancia de Wasserstein para distribuciones de probabilidad $W(q, p)$, dando lugar a la función objetivo

$$\min_G \max_{D \in \mathbb{D}} \mathbb{E}_{x \sim p_X(x)} \left[D(x) \right] - \mathbb{E}_{z \sim p_z(z)} \left[D(G(z)) \right],$$

donde $\mathbb{D}$ es el conjunto de funciones Lipschitz-1. Para hacer cumplir la restricción de que el Discriminador debe pertenecer a $\mathbb{D}$, sugieren truncar los pesos de D entre los valores $[-c, c]$. A esta alternativa se la conoce como Wasserstein GAN (WGAN) [1].

Gulrajani et al. toman el aporte previo, y agregan que esa manera de imponer la restricción de Lipschitz-1 puede traer algunas complicaciones, como llevar el gradiente a valores extremos o nulos, y desaprovechar la capacidad de la GAN limitando al discriminador a funciones muy simples. Proponen en cambio imponer la restricción de manera "suave", incorporándola directamente en la función objetivo, que ahora quedaría:

$$\mathbb{E}_{x \sim p_X(x)} \left[D(x|y) \right] - \mathbb{E}_{z \sim p_z(z)} \left[D(G(z|y)) \right] - \lambda \mathbb{E}_{\hat{x}} \left[\left(||\nabla_{\hat{x}} D(\hat{x}|y)||_2 - 1 \right)^2 \right] \quad (2.2)$$

El término de la derecha penaliza al discriminador cuando su gradiente respecto a los inputs (en nuestro caso las observaciones) se aleja de 1. $\hat{x}$ representa combinaciones convexas entre observaciones de $G(Z)$ y de X, y λ es un parámetro que determina con qué impacto entrará este término en la función. A esta alternativa se la conoce como Wasserstein GAN con Gradient Penalty (WGAN-GP) [4]. Es importante notar que 2.2 es la función de pérdida propuesta por estos autores, sólo que adaptada a nuestro problema, condicionando tanto al generador como al discriminador a las etiquetas y, en nuestro caso, el tiempo.

Algo a tener en cuenta es que, dado que tanto el Generador como el Discriminador tomarán como inputs una observación o ruido y el momento del

tiempo, normalizamos los datos para que el valor de la variable tiempo se mueva entre los valores que registran los otros inputs. Por ejemplo, nuestro generador podría estar tomando como input tanto a un vector de ruido con media cero como a un año que podría ser 1995.

Para finalizar, un punto interesante a tener en cuenta es el discutido por Zaheer et al. [13], quienes señalan la dificultad que presentan las GANs a la hora de querer aprender distribuciones unidimensionales simples, incluso tomando consideraciones como el uso de la distancia Wasserstein. En el artículo, los autores llegan a la conclusión que estos modelos se colapsan sobre la moda en estos casos y no logran generar muestras fieles a la distribución entera. Prestaremos atención a esto a la hora de evaluar el modelo sobre nuestros datos.

CAPÍTULO 3

Simulaciones e Implementación

Implementamos las tres alternativas y las testeamos en conjuntos de datos modelos, sabiendo antemano qué deberíamos esperar, para así evaluar su desempeño. Toda la implementación fue llevada a cabo en Python, con uso de paquetes de código abierto[3]. Para cada caso, intentamos predecir la evolución de distribuciones normales o mixturas de normales, de las cuales tomamos muestras de tamaño 1000 por cada período de tiempo. Tomamos 20 períodos como set de entrenamiento y buscamos predecir para los 5 siguientes. Para cada método visualizamos sus predicciones comparándolas con las distribuciones del período de muestra, para las cuales graficamos su estimación realizada con kernels gaussianos en base a las muestras que tomamos de las originales (las mismas muestras con las que se ajustan los procedimientos). Por último, evaluamos la calidad de las predicciones ajustando cada alternativa 100 veces y promediando la divergencia Kullback-Leibler [7] respecto a las densidades verdaderas. Intentamos utilizar las mismas configuraciones e hiperparámetros para cada caso. A continuación, describiremos en líneas generales la implementación de las tres alternativas para todas las simulaciones y el caso de aplicación, si bien luego haremos algunas modificaciones según el caso.

Paramétrico

Para cada período de los observados en la muestra ajustamos las distribuciones correspondientes sobre las observaciones generadas. Para el caso donde tenemos únicamente una moda ajustamos una distribución por período, mientras que para los casos con más de una moda ajustamos mixturas de distribuciones. Una vez realizadas las estimaciones, proyectamos las series de parámetros a futuro usando distintos modelos según el caso. Con estas proyecciones luego reconstruimos las distribuciones en $T + h$ y las evaluamos sobre la grilla.

FPCA

Como primer paso ajustamos sobre las muestras generadas una distribución para cada período mediante kernels utilizando la implementación de Scikit Learn, tomando como ancho de banda la regla de Silverman [12]. Luego, tomamos la log-densidad y la evaluamos sobre una grilla. Sobre esta discretización, re-expresamos las log-densidades en una base de BSplines. El número de BSplines

[3]Código disponible en https://github.com/lipinoelbreve/densities-time-series

fue elegido como la mínima cantidad tal que las funciones se sigan asemejando visualmente a las originales. Para todos los casos elegimos un número de 30 BSplines. Luego, aplicamos FPCA llevando las funciones a una base de K funciones ortogonales. En general, tomando $K = 2$ logramos explicar más de un 99 % de la variabilidad de las funciones, si bien en algunos casos decidimos tomar valores de K mayores para mejorar la calidad de los resultados.

Una vez ajustadas las autofunciones, obtenemos los *scores* para las mismas y con ellos construimos las series que proyectamos a futuro con tendencia lineal, polinómica o exponencial, o bien con un modelo ARIMA según el caso para reconstruir las log-densidades en $T + h$. El paso final es aplicar exponenciación a las funciones y normalizar sus integrales a 1.

GAN

Para la GAN buscamos utilizar arquitecturas lo más parsimoniosas posibles en cantidad de capas y nodos en comparación con las arquitecturas usuales que se encuentran disponibles[4], entendiendo que las mismas buscan entrenarse para dominios mucho más complejos que los que enfrentaremos nosotros. Terminamos empleando un Generador con cuatro capas ocultas completamente conectadas, de 16 nodos cada una, y activación Leaky ReLU a excepción de la última (Figura 3.1). El vector de ruido es unidimensional (un escalar), muestreado de una normal $\mathcal{N}(0, 1)$.

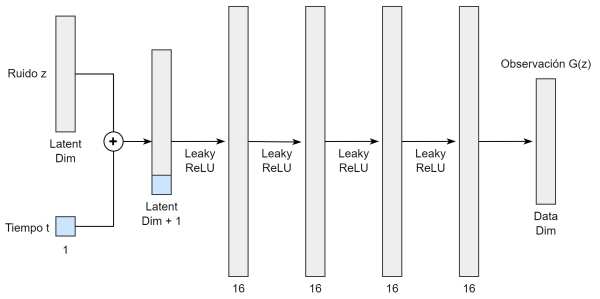


Figura 3.1: Modelo Generador

En cuanto al Discriminador (Figura 3.2), el mismo tiene dos capas ocultas completamente conectadas, de 64 nodos cada una, con dropout y activación Leaky ReLU a excepción de la última, la cual devuelve un escalar con función de activación sigmoide, representando la probabilidad de que la observación de input sea verdadera.

Respecto a los hiperparámetros, mantuvimos el estándar utilizado en implementaciones previas; en particular, utilizamos un optimizador Adam con $\beta_1 = 0{,}5$, $\beta_2 = 0{,}999$ y *learning rate* de $2e - 4$ para ambas redes. Respecto al entrenamiento, fijamos $ncritic = 5$ como la cantidad de *steps* entre cada actualización del generador. El único hiperparámetro para el cual nos desviamos de los valores usuales es λ en la función de pérdida 2.2, que suele fijarse en 10 en

[4]Para la implementación del modelo nos basamos en el código disponible en https://github.com/eriklindernoren/PyTorch-GAN haciendo los cambios necesarios para adaptarlo a nuestro problema.

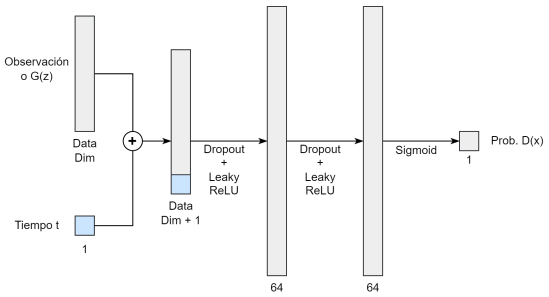


Figura 3.2: Modelo Discriminador

las implementaciones previas [4]. En nuestro caso obtuvimos mejores resultados con valores menores, tan bajos como 0,5.

Al entrenar las GANs, el criterio utilizado para determinar el corte del entrenamiento fue cuán similares visualmente a las distribuciones reales resultaban las distribuciones que el generador iba produciendo para distintos momentos del tiempo. En general, dejamos que el modelo se entrene hasta las 2000 épocas. Si bien en algunos casos se podría haber entrenado por menos épocas sin grandes pérdidas en desempeño, preferimos mantener la duración del entrenamiento igual para todos los casos.

Una vez entrenado el modelo, el último paso consiste en reconstruir las funciones para $T+h$. Para esto, muestreamos un vector de ruido que el generador toma como input junto a las etiquetas correspondientes a los períodos h que queremos generar. El generador nos devuelve las muestras condicionales a cada momento, sobre las cuales ajustamos las distribuciones $\hat{f}_{T+h}$ utilizando kernels.

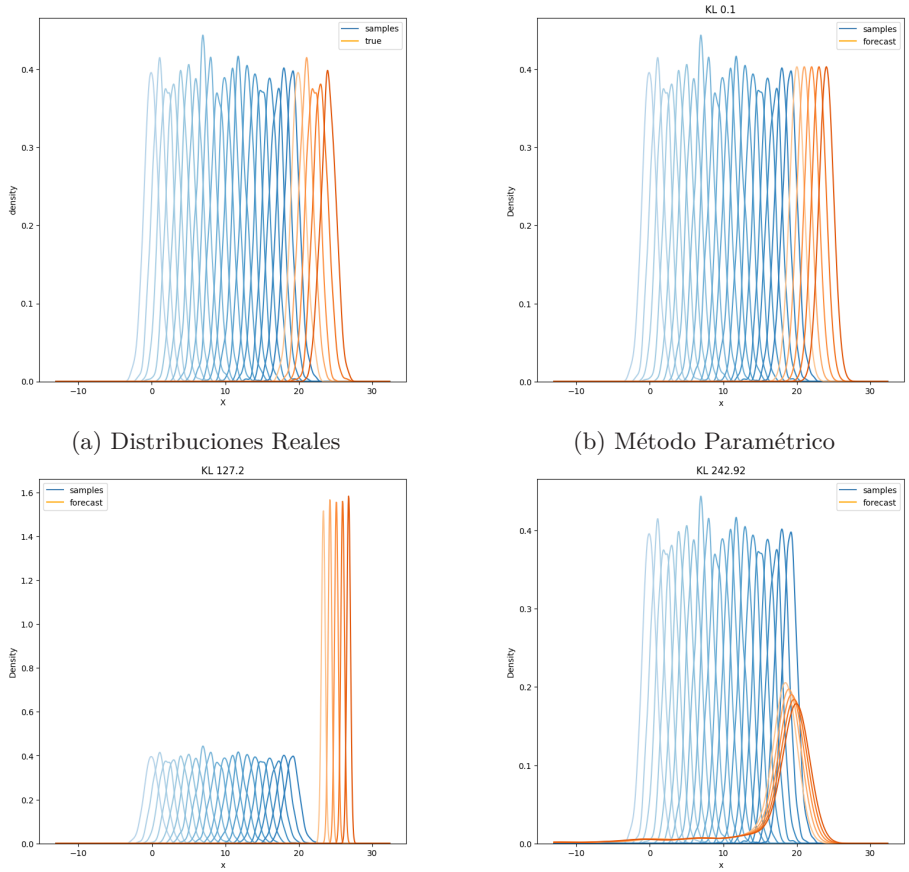
3.1. Desplazamiento Uniforme

El primer caso que simulamos es el de una normal con desvío 1 en una dimensión, cuya media va aumentando uniformemente de a una unidad con el paso del tiempo, según la Figura 3.3a. Las alternativas deberían capturar este desplazamiento y continuarlo en el tiempo.

Para el caso paramétrico ajustamos una normal para cada período utilizando la implementación de Scipy, y luego proyectamos las medias con una tendencia lineal y dejamos los desvíos constantes en su media del período de muestra. Luego de un breve ajuste, el método logra los resultados que esperábamos, como se ve en la Figura 3.3b con una divergencia Kullback-Leibler (KL) promedio de 0,1 respecto a las densidades verdaderas. Recalcamos que este buen resultado se debe a que conocemos a priori la distribución que genera los datos, y por esa razón buscamos ajustar normales.

Para el método de FPCA hacemos el ajuste con la implementación de la librería Scikit-FDA. Utilizamos dos autofunciones cuyos *scores* proyectamos a futuro con una tendencia lineal y polinómica de grado 2, respectivamente. Tras un breve ajuste, el método logra detectar el desplazamiento de las distribuciones, aunque exagera el salto entre el período de muestra y el período de predicción, y subestima la dispersión de los datos (Figura 3.3c). El ajuste es peor que el del método paramétrico, tanto visualmente como en términos cuantitativos, con un KL promedio de 127,2.

Respecto a la GAN, entrenamos al modelo por 2000 épocas con los parámetros descritos arriba utilizando la librería Pytorch. Esta fue la alternativa que más tiempo nos llevó en entrenar, si bien podríamos haber cortado el entrenamiento más temprano. Si con el procedimiento de FPCA teníamos una exageración en el desplazamiento, con la GAN observamos más bien una subestimación (Figura 3.3d), quedando las predicciones para $T+h$ superpuestas con las densidades del período de muestra. Además, las densidades presentan una cola pesada hacia la izquierda que no está presente en las distribuciones de muestra. De las tres alternativas, produce el peor ajuste con un KL promedio de 242,92.



(a) Distribuciones Reales

(b) Método Paramétrico

(c) Método FPCA

(d) Método GAN

Figura 3.3: Resultados - Tendencia Lineal. El mejor ajuste es el realizado por el método paramétrico, en gran medida porque ajustamos normales sabiendo de antemano que esa es la distribución verdadera de los datos.

3.2. Desplazamiento Uniforme con dos Modas

Similar al ejemplo anterior, ahora buscamos ver si los métodos logran capturar el desplazamiento de la distribución cuando la misma consiste en dos

normales separadas una de la otra. De nuevo, tenemos 20 períodos de muestra y 5 para predecir; de las 1000 observaciones de cada período, 500 provendrán de una normal con desvío 1 y 500 de una distribución idéntica pero desplazada en 10 unidades, como se ve en la Figura 3.4a.

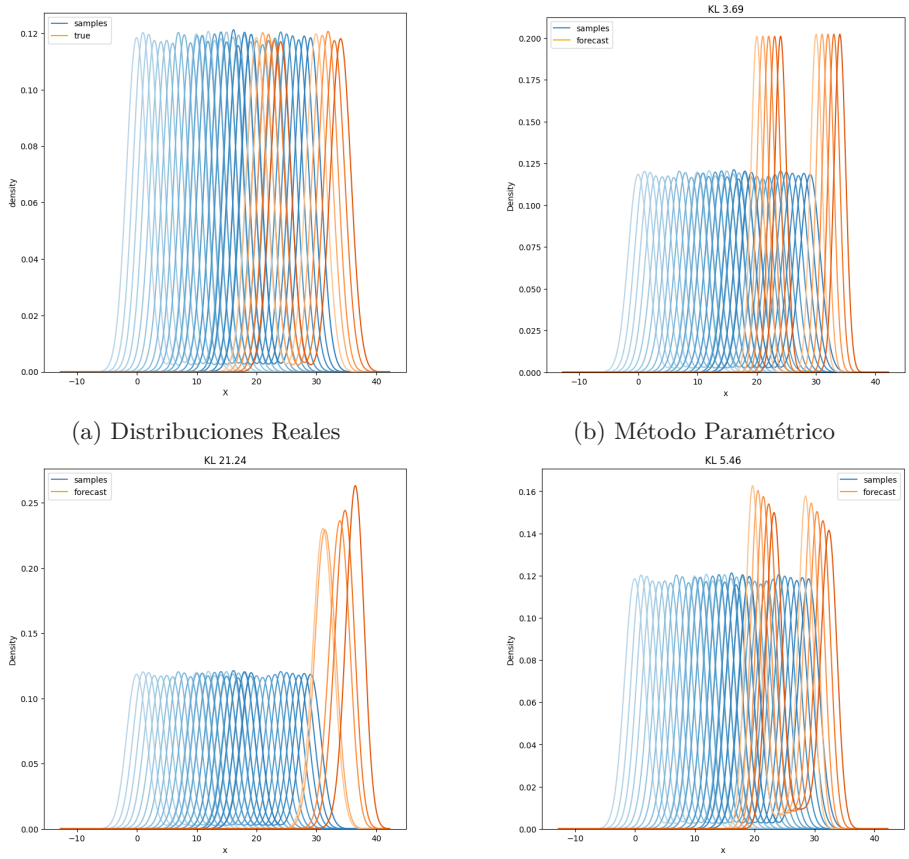
Para el caso paramétrico, al tener ahora dos modas, ajustamos una mixtura de dos normales por período estimando los parámetros (media y desvío de ambas) y los pesos de cada una, utilizando la implementación de la librería *pomegranate*. Una vez estimados los parámetros, proyectamos únicamente las medias a futuro utilizando un modelo ARIMA para cada media. Podríamos haber utilizado una tendencia lineal, pero preferimos usar ARIMA ya que los resultados no difieren y este último modelo también lo utilizaremos más adelante para otras series que no son tan sencillas. Los desvíos y las ponderaciones de las normales en las mixturas los dejamos constantes como su media en el período de muestra. Al igual que en el caso de una única moda, las predicciones son correctas capturando en buena medida el desplazamiento de ambas modas (Figura 3.4b). El único defecto a la vista es que la dispersión de las distribuciones es menor a la del período de muestra. También, al igual que con una moda, el método paramétrico logra el mejor ajuste con un KL promedio de 3,69.

Para el método de FPCA, repetimos los mismos pasos que en el caso de una moda ajustando dos autofunciones, solo que ahora proyectamos los *scores* con un modelo ARIMA. A la hora de predecir para $T + h$ notamos resultados significativamente peores que con el método paramétrico con un KL promedio de 21,24 (Figura 3.4c). De las dos modas, el ajuste sólo logra capturar el desplazamiento de la mayor, mientras que la moda menor se pierde por completo en la predicción. Realizamos pruebas con distintas especificaciones del modelo ARIMA y ajustando distintos números de autofunciones, pero en ningún caso logramos obtener un desplazamiento para ambas modas. Sospechamos que en parte esto se debe a que, al igual que en el desplazamiento de una única moda, estamos esperando que el método genere distribuciones cada vez más alejadas del dominio sobre el que se entrenó. Hay que tener en cuenta que este método está pensado para series estacionarias de funciones que se encuentran definidas en un intervalo compacto, como detallamos en la Sección 2.2; por la naturaleza de la simulación que construimos, las densidades continuarían desplazándose indefinidamente, quedando eventualmente fuera del intervalo donde están definidas las f_t, rompiendo el supuesto de estacionariedad. Este método, entonces, no sería el adecuado si los datos no son estacionarios, pudiendo dar lugar a desajustes como los que observamos.

Por último, el modelo de redes adversarias es entrenado sin cambios respecto al caso de una moda. Luego de entrenarse por 2000 épocas, el modelo muestra resultados alentadores identificando el desplazamiento para ambas modas e incluso acertando en gran medida a la dispersión de las distribuciones (Figura 3.4d). Esta vez, el modelo de redes supera al procedimiento de FPCA con un KL promedio de 5,46.

3.3. Cruce de Dos Modas

La siguiente prueba que haremos es similar a la anterior, sólo que ahora las modas comenzarán separadas, se juntarán cada vez más con el paso del tiempo, hasta que en un punto se cruzarán y pasarán a alejarse. Los métodos deberían



(a) Distribuciones Reales

(b) Método Paramétrico

(c) Método FPCA

(d) Método GAN

Figura 3.4: Resultados - Tendencia Lineal con Dos Modas. Nuevamente el método paramétrico logra el mejor ajuste. Esta vez, el modelo de redes tiene un mejor desempeño que el ajuste por FPCA, el cual sólo detecta una moda.

capturar esto y continuar separándolas una de la otra. Al igual que en el caso anterior, ambas normales tienen desvío 1 y de cada una de ellas muestreamos 500 observaciones por período (teniendo 1000 en total por período). La diferencia es que la moda menor se irá desplazando hacia la derecha y viceversa, para terminar en lugares opuestos en T (Figura 3.8a).

Para el caso paramétrico volvemos a ajustar, para cada período, una mixtura de dos normales y proyectamos sus parámetros a futuro. Algo a tener en cuenta, y que sólo sucede con este método, es que tendremos que hacer un supuesto sobre la identificación de cada componente de la mixtura en cada momento. En otras palabras, en cada período ajustamos $g_{1,t}(x)$ y $g_{2,t}(x)$ que, combinándolas con los pesos $w_{1,t}$ y $w_{2,t}$, dan lugar a nuestra estimación de $f_t(x)$; pero el problema es que g_1 y g_2 no están ordenadas, y cuando las estimemos deberemos ordenarlas de alguna forma, para luego poder construir la serie de tiempo de los parámetros a proyectar. En el caso de la Sección 3.2 esto no parecía algo tan grave - teníamos dos modas igualmente separadas en cada momento, que

se iban desplazando a la par[5]. Sin embargo, ahora sí es necesario hacer algún supuesto, ya que en un momento las modas se cruzan. Lo más razonable sería o bien pensar que hay una moda que cae y otra que sube (lo que realmente está pasando), o bien que las modas comienzan a juntarse y luego se vuelven a separar. En términos de las series de tiempo de las medias, esto implicaría alguno de los casos señalados en la Figura 3.5:

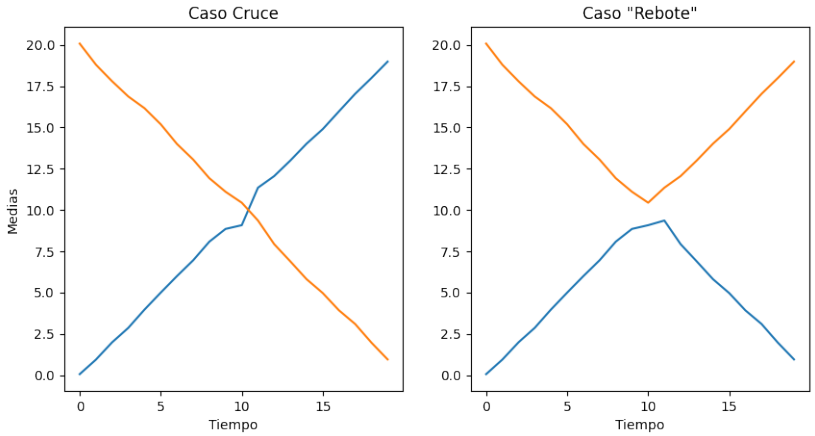


Figura 3.5: Distintas asignaciones llevan a distintas series de tiempo.

En este caso esto tampoco parecería ser un problema ya que, de cualquier forma, para $T+1$ estaremos diciendo que una moda aumentó y la otra cayó. Sin embargo, para otro tipo de funciones donde necesitemos estimar mixturas de más componentes o donde estos no tengan un comportamiento tan estable en el tiempo, el problema de la identificación podría cobrar relevancia. Esta vez vamos a suponer el primer caso, es decir, que las modas se cruzan.

Retomando, luego de hacer este supuesto, proyectamos las medias de ambas normales con ARIMA. Volvemos a dejar los desvíos y las ponderaciones constantes como la media en el período de muestra. Al evaluar las funciones estimadas para $T+h$, el ajuste logra identificar el alejamiento de ambas normales de manera correcta, si bien nuevamente subestima su desvío (Figura 3.8b), dando lugar a un KL promedio de 16,87.

Con el método funcional, en este caso puntual, no logramos reconstruir las funciones para $T+h$ utilizando únicamente 2 componentes. Las series de tiempo de los *scores* se vuelven más complicadas a la hora de agregar componentes cada vez menos significativos. Reconocemos que debimos experimentar con varias especificaciones de ARIMA sobre las series e incluso dejar algunas series como constantes para obtener resultados razonables. Para esto, fue necesario analizar visualmente las autofunciones e interpretar qué estaba capturando cada una.

Debemos tener en cuenta que las autofunciones que vemos en la Figura 3.6 están basadas en las log-densidades, por lo que valores negativos en esta escala se traducirían en valores entre 0 y 1 en la escala original de las funciones. La primera autofunción, la que mayor variabilidad explica, determinaría cuán centradas en el dominio de las muestras estarán las funciones. Analizando su

[5]Podría no ser el caso. ¿Cómo saber si en el fondo son dos poblaciones que se van alternando la una con la otra?

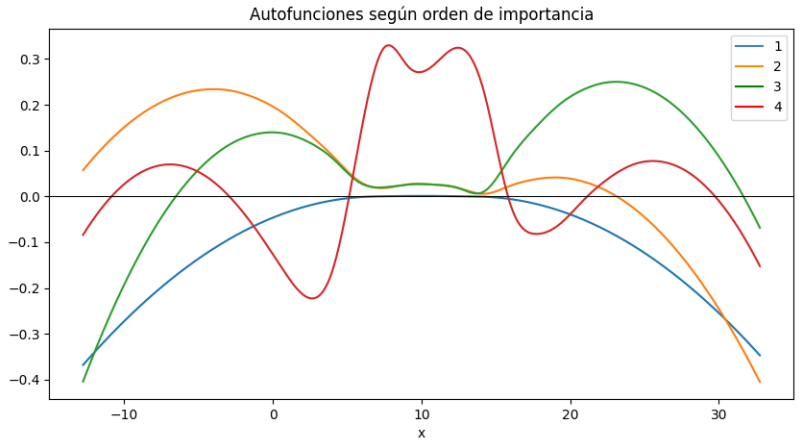


Figura 3.6: Autofunciones para el cruce de modas en orden de importancia. Escala logarítmica.

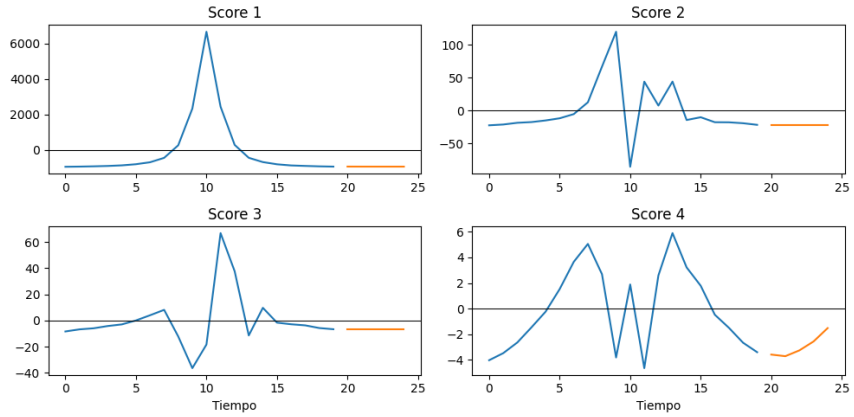


Figura 3.7: Scores para el cruce de modas. En azul el período de muestra; en naranja las proyecciones para $T + h$.

serie de *scores* en la Figura 3.7, vemos que al comienzo y al final del período se la pondera por valores negativos, dándole más importancia a los valores hacia los extremos del dominio. Se vuelve positiva hacia la mitad del período, cuando las dos modas se cruzan en el centro. La continuación de esta serie parecería correcta en valores negativos y con una ligera tendencia a valores menores, indicando que las series se alejarán cada vez más del centro. Para eso ajustamos una tendencia exponencial únicamente en la parte descendente de la serie.

La segunda y tercera autofunción regulan los extremos del dominio, cada una favoreciendo un extremo y castigando el otro. Las proyecciones de ARIMA dejan ambas series relativamente constantes; de todas formas, tuvimos que forzar a la segunda a que quede constante en el último valor registrado, ya que ARIMA la sobrestimaba, haciendo que las predicciones tuvieran más peso hacia la izquierda.

La cuarta autofunción le da importancia nuevamente al centro del dominio

y castiga, en parte, los extremos. Sin embargo, este componente aporta muy poco a la reconstrucción de las funciones (notar la magnitud de los *scores* en comparación con los de la primera autofunción). Decidimos incorporarla ya que mejoraba ligeramente los resultados, pero se vuelve difícil de interpretar la función y su serie de *scores*.

Con estas consideraciones logramos que las densidades para $T + h$ del método FPCA tengan una apariencia razonable pero, de todas formas, no se logra capturar del todo bien el desplazamiento, quedando todas las proyecciones en el mismo lugar (Figura 3.8c). Para esta simulación puntual, con un KL de 14,13, FPCA supera al método paramétrico, aunque no está capturando tan bien el desplazamiento.

Por último, para el método de GAN entrenamos el modelo exactamente igual que en los casos anteriores y, a juzgar por los resultados en el caso del cruce de dos modas, es la alternativa que mejor logró capturar el cambio de las distribuciones en el tiempo y continuarlo a futuro, con un KL de 10,51 (Figura 3.8d). A pesar de esto, no logra separar completamente bien las modas, asignando una alta densidad a los valores entre ambas (cuando dicha región debería tener una densidad cercana a cero). Además produce una asimetría que no está presente en las muestras, sobre todo para la moda mayor.

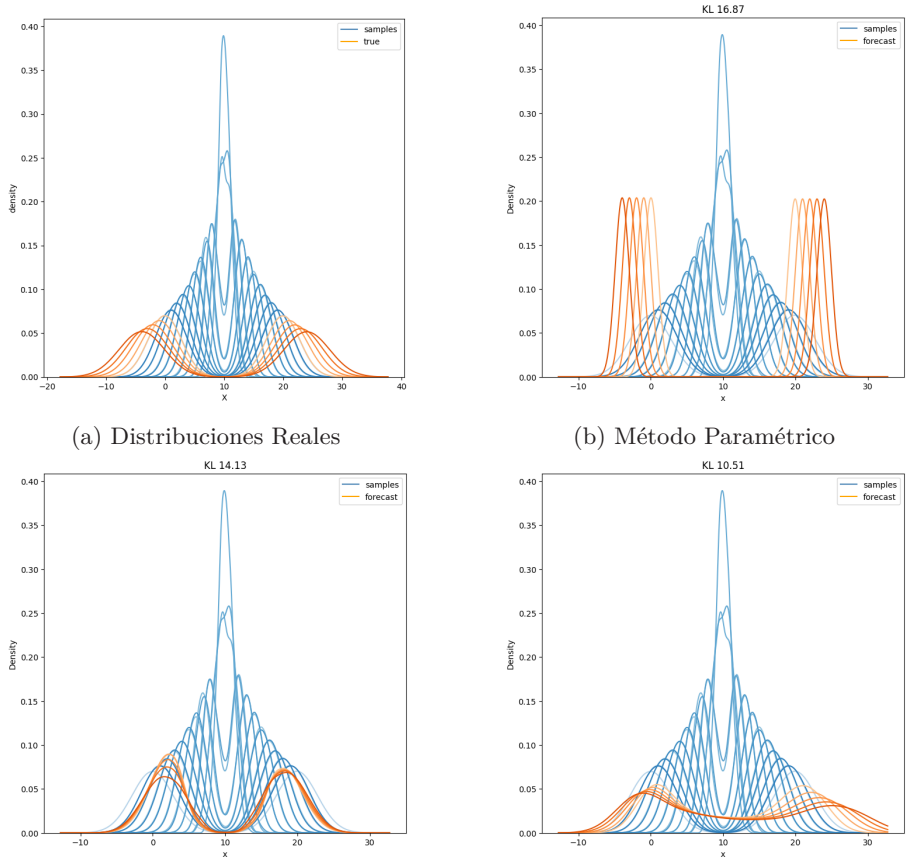
3.4. Cambio de Moda

Para la última prueba que realizamos, en lugar de tener modas que se desplazan, las mismas estarán siempre en la misma posición cambiando en el tiempo su proporción. Lo que hacemos es generar 1000 muestras tomando una fracción con una moda menor y la otra fracción con una mayor. Para cada período la fracción de una subpoblación aumenta en detrimento de la otra (Figura 3.9a).

Para el método paramétrico volvemos a ajustar una mixtura de dos normales para cada período. La principal diferencia respecto a los ejemplos anteriores es que ahora modelaremos la serie de tiempo de las ponderaciones w_0 y w_1 además de las medias. Al tratarse de ponderaciones entre 0 y 1, decidimos modelar esta serie como una tendencia lineal en el tiempo del *logit* de w_0, y luego tomamos w_1 como el complemento de w_0. Para las medias volvemos a utilizar modelos ARIMA, aunque podríamos haber tomado una tendencia lineal o incluso dejarlas constantes ya que, en este caso, no varían en el tiempo.

Al proyectar las funciones para $T + h$ con el método paramétrico notamos, en la Figura 3.9b, que se captura correctamente el cambio desde la moda menor hacia la mayor, si bien las densidades quedan fuera de escala al estar subestimando nuevamente sus desvíos, dando lugar a un KL de 5,0.

En cuanto al método no paramétrico de FPCA decidimos ajustar tres autofunciones. Si bien la primera explica por sí misma en gran medida la ponderación de las modas, incorporamos las otras dos autofunciones ya que así logramos que las densidades para $T + h$ no sólo reflejen el paso de una moda hacia la otra, sino que también hace se acomoden respecto a las distribuciones en la muestra. En líneas generales, observando la Figura 3.9c, notamos que la proyección de las funciones es fiel a las verdaderas, únicamente con el detalle de que la moda menor es castigada un poco más de lo debido. De las tres



(a) Distribuciones Reales

(b) Método Paramétrico

(c) Método FPCA

(d) Método GAN

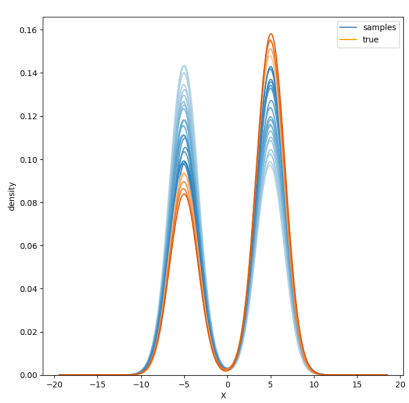
Figura 3.8: Resultados - Cruce de Modas. Esta vez el modelo de redes resulta ser el mejor en capturar el desplazamiento de ambas modas, si bien produce una asimetría que no estaba presente en las densidades originales.

alternativas, este procedimiento logra el mejor ajuste con un KL de 0,33 para esta simulación.

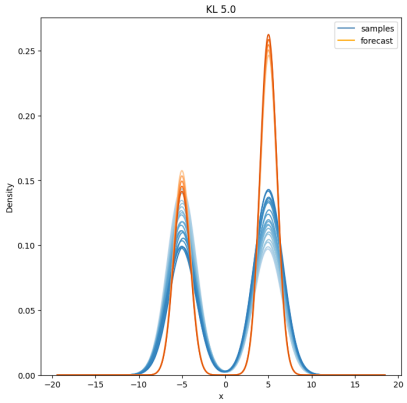
Por último, el método de GANs condicionales logra aprender a grandes rasgos el cambio en las funciones sin necesidad de modificar hiperparámetros y entrenando con la misma cantidad de épocas que en los casos anteriores. De todas formas, notamos algunos defectos en el ajuste, como por ejemplo un ligero desplazamiento de las modas respecto a las originales (Figura 3.9d), ubicando al método en segundo lugar con un KL de 3,68.

3.5. Resultados

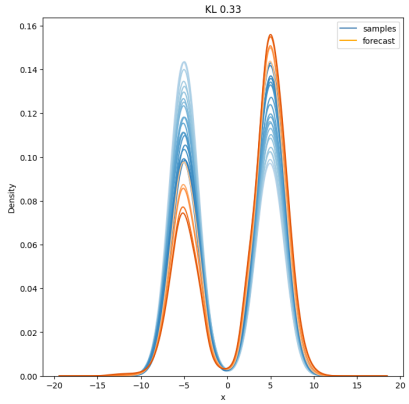
A continuación, presentamos los resultados tras 100 ajustes de cada alternativa sobre cada set de datos. Sólo realizamos 100 réplicas debido a que el procedimiento es costoso computacionalmente, en especial el entrenamiento de la GAN.



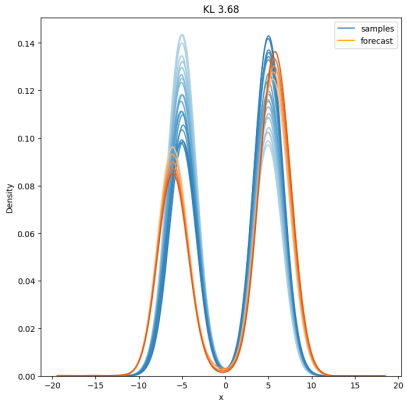
(a) Distribuciones Reales



(b) Método Paramétrico



(c) Método FPCA



(d) Método GAN

Figura 3.9: Resultados - Cambio de Modas. El método FPCA produce el mejor ajuste, con densidades proyectadas muy similares a las reales. Le sigue de cerca el modelo de redes.

Para mostrar los resultados, en cada iteración generamos nuevas muestras para los cuatro escenarios descritos en las secciones anteriores y ajustamos, en cada caso, los tres procedimientos usando exactamente los mismos parámetros y configuraciones. Por ejemplo, para FPCA ajustamos la misma cantidad de autofunciones y utilizamos los mismos modelos para las series de tiempo, y para las redes adversarias usamos también los mismos parámetros y épocas de entrenamiento.

A continuación presentamos el promedio de las divergencias Kullback-Leibler entre las densidades predichas y las reales, tomado para las 100 réplicas.

Método	**1 Moda**	**2 Modas**	**Cruce de Modas**	**Cambio de Moda**
Paramétrico	**0,11**	**3,62**	16,55	4,91
FPCA	92,84	22,95	14,03	**0,20**
WGAN	78,93	9,57	**9,42**	3,31

Tabla 3.1: Divergencias Kullback-Leibler promedio.

La Tabla 3.1 muestra que para el desplazamiento de una y dos modas, el mejor método resultó ser el paramétrico, como ya habíamos comprobado visualmente. En ambos casos, al margen del método paramétrico, el modelo de redes adversarias mostró mejores resultados que el procedimiento de FPCA. Creemos que esto se debe a que en ambas simulaciones las series de densidades no eran estacionarias, y las predicciones debían hacerse cada vez más lejos del dominio en el que se encontraban las muestras, situación que no es ideal para el método de FPCA según lo que comentamos en la Sección 3.2.

Para los casos de cruce de modas y cambio de modas, las alternativas que mejor lograron capturar dichos comportamientos fueron la de redes adversarias y la de FPCA, respectivamente. Cabe destacar que, si bien es el más lento en entrenar, el modelo de redes fue exactamente el mismo para todos los casos, mientras que los métodos paramétrico y de FPCA requirieron siempre un cuidado a la hora de ajustar densidades y proyectar series de tiempo. Además, el modelo de redes nunca resultó ser el peor de los tres en ninguno de los cuatro casos que planteamos; el set de datos donde presentó mayores dificultades fue el desplazamiento de una única moda, lo que nos llama la atención ya que a priori parecería ser un caso más simple de predecir que el resto. Así y todo, en dicho escenario, logró un mejor desempeño que la alternativa de FPCA.

CAPÍTULO 4

Caso de Aplicación

En esta sección aplicamos los tres métodos al set de datos que fue el disparador de este trabajo. Se quería hacer un análisis sobre la evolución de las características de los satélites que se lanzaron en los últimos años para un proveedor de servicios en órbita. Algunos ejemplos de estos servicios son: brindar cobertura de Internet a estos satélites, remoción de escombros, repostaje en órbita, suministro eléctrico, remolque, entre otros. En concreto, un proveedor de remoción de escombros es aquel que se encarga de eyectar de órbita a satélites obsoletos o dañados, para liberar espacio y evitar colisiones con otros satélites. Esto se puede llevar a cabo con vehículos con un sistema de propulsión propio que se acoplarían al satélite a remover y lo impulsarían fuera de órbita. Una pregunta que surge al desarrollar este vehículo es respecto a la masa de los satélites que tendrá que remover para evaluar, por ejemplo, qué sistema de propulsión desarrollar o adquirir; el vehículo presentará características distintas si planea brindar servicio a satélites que rondan en las decenas de kilos o a satélites que están en el rango de toneladas.

Una primera aproximación que se podría hacer para responder esta pregunta, es estudiar la evolución de alguna medida de centralidad para la masa de los satélites lanzados año a año. Así se podría ver, por ejemplo, si el satélite promedio se va tornando más pesado o más liviano con el paso del tiempo, y proyectar esta medida a futuro para los años que se planee brindar servicio. Sin embargo, esta medida de centralidad podría ocultar información útil, como un subgrupo del mercado que se esté moviendo en contra de la media y que podría ser un nicho que el proveedor se perdería de aprovechar. Para hacer un análisis más abarcativo, entonces, se podría estudiar la distribución entera de la masa de los satélites lanzados y proyectar su evolución. Con estas distribuciones proyectadas, no sólo se podría recuperar cualquier medida de centralidad para hacer un análisis como el del ejemplo anterior, sino que se podría contar con la distribución entera, permitiendo hacer un análisis más profundo. Por ejemplo, se podrían utilizar las distribuciones para muestrearlas y hacer simulaciones, algo que antes no era posible.

Los datos que utilizamos fueron provistos por Teri Grimwood de la Union of Concerned Scientists (UCS). Citando la página de la UCS[6], la misma provee un set de datos con "detalles de los satélites que orbitan la Tierra actualmente, incluyendo su país de origen, propósito, y otros detalles operacionales". Estos detalles operacionales incluyen características de las órbitas de los satélites

[6] https://www.ucsusa.org/resources/satellite-database

(variables como apogeo, perigeo o inclinación), y la masa de los mismos, variable que es de nuestro interés en este caso.

El set de datos que ofrecen es público, con el detalle de que se limita a los satélites que orbitan actualmente la Tierra. Es decir que, un satélite lanzado en el pasado y que ya no se encuentre en órbita, no estará disponible en la muestra. Es por esto que le pedimos a la UCS que nos diera acceso a sets de datos publicados en el pasado, los cuales pusieron a nuestra disposición y con ellos pudimos obtener una muestra más representativa de los satélites lanzados en los últimos años.

El set de datos con el que contamos parte de 7137 observaciones, es decir, 7137 satélites lanzados a órbita desde 1967 hasta 2022. De este set original, eliminamos primero las observaciones para las cuales no contamos con la variable *launch_mass* (masa al momento de lanzamiento en kilos), quedándonos con 6827 observaciones.

Algo a tener en cuenta en este punto es que los satélites y sus órbitas pueden ser muy distintos entre sí. En particular, una separación grande que se puede hacer es entre aquellos que se encuentran en órbitas terrestres bajas (*Low Earth Orbit* o LEO) y órbitas terrestres geoestacionarias (*Geostationary Orbit* o GEO). Las órbitas LEO se encuentran por debajo de los mil kilómetros de altitud, y es el destino más popular para satélites comerciales con fines como la observación terrestre o telecomunicaciones. Estos satélites suelen ser pequeños, en general concentrándose casi todos por debajo de los 100 kilos (Figura 4.1a), y orbitan la Tierra cada 90 minutos aproximadamente.

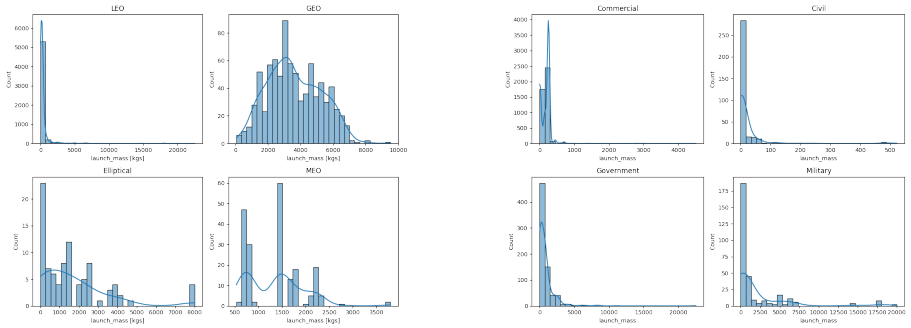
Por otro lado, las órbitas GEO se ubican a 36.000 km de altitud. Los satélites lanzados a GEO suelen ser más complejos, costosos y pesados (hasta 10.000 kilos), haciendo de esta una órbita menos popular para el sector privado. Un satélite en GEO orbita la Tierra cada 24 horas, por lo que desde la perspectiva de un observador terrestre, un satélite en GEO estaría siempre en el mismo lugar (de allí el nombre geoestacionario), haciéndolos ideales para servicios de meteorología, navegación (GPS) y comunicaciones.

Al separarse tan marcadamente estos grupos de observaciones, decidimos acotar nuestro estudio a una de estas familias, quedándonos con los satélites lanzados a LEO por ser los más comunes (representan un 83 % de nuestros datos).

Dentro del grupo de LEO vemos que, según el tipo de usuario, los satélites también registran distintas distribuciones en sus masas. En la Figura 4.1b los satélites comerciales y civiles en casos muy raros superan los 500 kilos. Por otro lado, son varios los lanzados por usuarios gubernamentales que alcanzan los 5.000 kilos, y otros tantos de fines militares que registran hasta 20.000 kilos. Optamos por descartar las observaciones por encima de los 4.000 kilos, al ser una fracción muy pequeña de la muestra (alrededor de un 1 %) que al margen de estar en LEO no se relacionan con los otros satélites (dentro del grupo que descartamos había módulos de estaciones espaciales como la china[7], y otros satélites militares que muestran un comportamiento distinto al resto de las observaciones).

Volviendo a nuestro análisis, el siguiente paso es agrupar las observaciones en ventanas de tiempo. Elegimos agrupar las observaciones por el año de lanzamiento, y descartamos los años para los cuales teníamos pocos datos

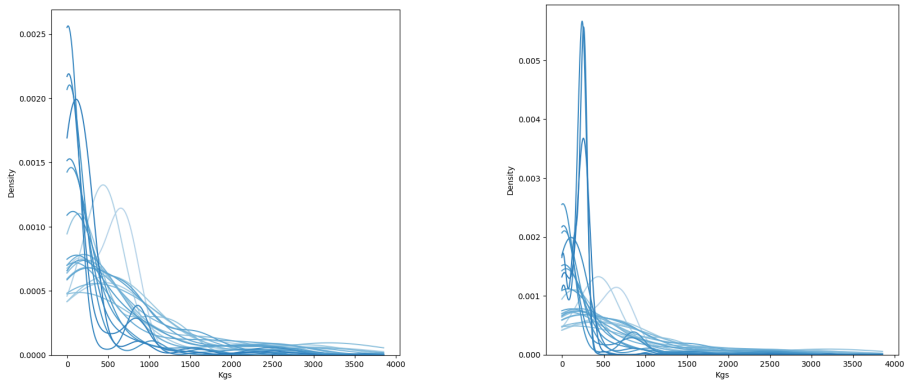
[7] https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2021-035A



(a) Según clase de órbita (b) Según usuario (para órbitas LEO)

Figura 4.1: Distribuciones de masa. La separación más importante es entre LEO con satélites principalmente debajo de los 1000 kilos, y GEO con alta densidad incluso hasta los 8000 kilos. Dentro de LEO, los principales operadores son de tipo comercial y civil, donde la concentración hacia satélites livianos es aún más marcada.

(arbitrariamente tomamos un umbral de 10 observaciones), teniendo en cuenta que el siguiente paso es ajustar una función de densidad para cada ventana de tiempo. De esta forma y resumiendo, nos quedamos con los satélites lanzados a LEO desde 1998 a la fecha por debajo de los 4.000 kilos y para los cuales contamos con su masa.



(a) 1998-2019 (b) 1998-2022

Figura 4.2: Distribuciones de masa anuales. En los últimos años se observó una fuerte concentración en torno a los 200 kilos, impulsada por el lanzamiento de constelaciones como Starlink.

Mediante una inspección visual de los datos, en la Figura 4.2 vemos que en general LEO se concentró siempre por debajo de los 1000 kilos. Lo que más llama la atención es el marcado pico centrado entre los 0 y 500 kilos, que se acentúo aún más en los últimos 3 años. Esto se debe a que, de los 5538 datos remanentes, 2391 pertencen a SpaceX y fueron lanzados entre 2019 y 2022, con la particularidad de ser todas observaciones prácticamente idénticas (entre 230

y 260 kilos[8]), formando parte de la constelación de Starlink[9].

Una vez acotada la muestra y teniendo una noción sobre cómo se ven nuestros datos, pasamos a aplicar los tres métodos descritos arriba. Nos quedamos con las distribuciones entre 1998 y 2021 como set de entrenamiento, y dejamos a 2022 como test.

Lo primero que hacemos es generar nuestras observaciones funcionales, estimando en base a los datos puntuales una función de densidad para cada año. Para esto, realizamos una estimación con kernel gaussiano utilizando la regla de Silverman [12] para el ancho de banda. Se debe notar que estas funciones serán la entrada del procedimiento de FPCA; tanto el método paramétrico como el método de redes tomarán como entrada las observaciones puntuales. Luego de ajustar cada método, compararemos la función de densidad predicha para 2022 con la ajustada en base a las observaciones reales.

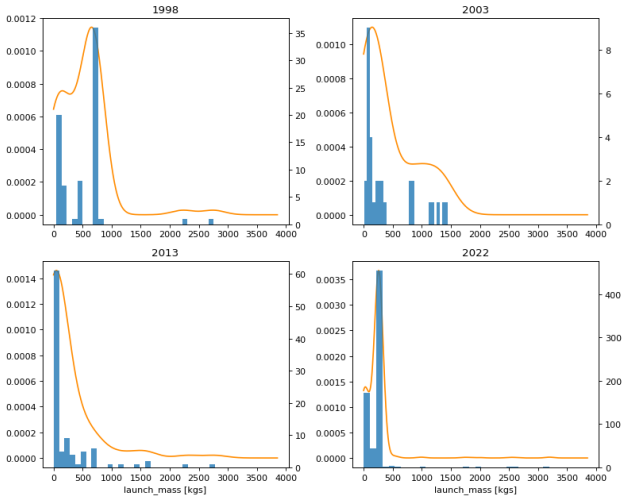


Figura 4.3: Distribuciones estimadas para algunos años en la muestra.

4.1. Método Paramétrico

Para el método paramétrico elegimos ajustar una mixtura de dos lognormales para cada período. La elección de la lognormal se debe en parte a que nuestras densidades tienen soporte en los reales positivos y en parte a que visualmente se asemejan a esta distribución, en el sentido de que presentan un pico para valores cercanos a cero y una cola hacia la derecha de la distribución. La decisión de utilizar una mixtura de dos lognormales en lugar de una sola se tomó para dar mayor flexibilidad al ajuste y porque, tras una inspección visual, las distribuciones suelen mostrar un grupo primario de satélites de poca masa, y otro grupo menor en cantidad de observaciones de satélites más pesados.

Tras ajustar las mixturas para cada año, nos quedamos con una serie de tiempo de 5 parámetros (dos medias, dos desvíos y la ponderación de una de las mixturas). Suponemos que los desvíos quedarán constantes y para $T + h$ los

[8]https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2019-074D
[9]https://www.starlink.com/technology

tomamos como su media del set de entrenamiento. Las medias las proyectamos con un modelo ARIMA y, para el peso de la mixtura, ajustamos un modelo lineal generalizado con función de enlace *logit* con el tiempo como variable independiente.

La divergencia Kullback-Leibler entre la función predicha y la ajustada en base a los datos es de 0,27. En la Figura 4.7a vemos que la densidad estimada acierta en concentrarse en torno a valores cercanos al cero, si bien no logra capturar del todo el pico alrededor de los 200 kilos detrás del cual se encuentra la constelación de Starlink.

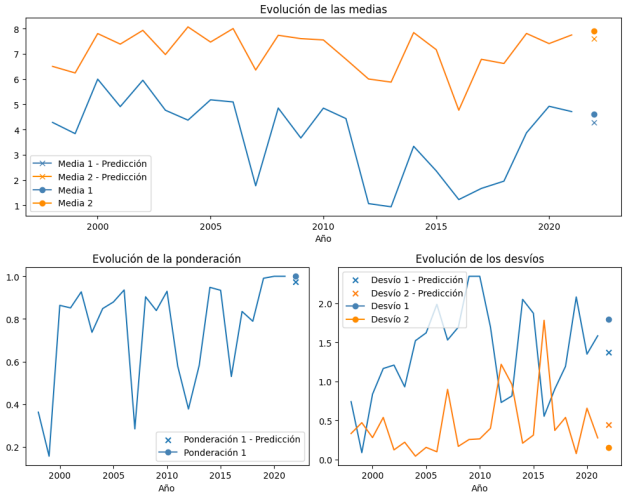


Figura 4.4: Evolución de los parámetros. Las medias son modeladas con ARIMA; la ponderación es proyectada con un modelo linear generalizado con función de enlace *logit* y el tiempo como variable independiente; los desvíos son tomados como su media en los períodos de muestra.

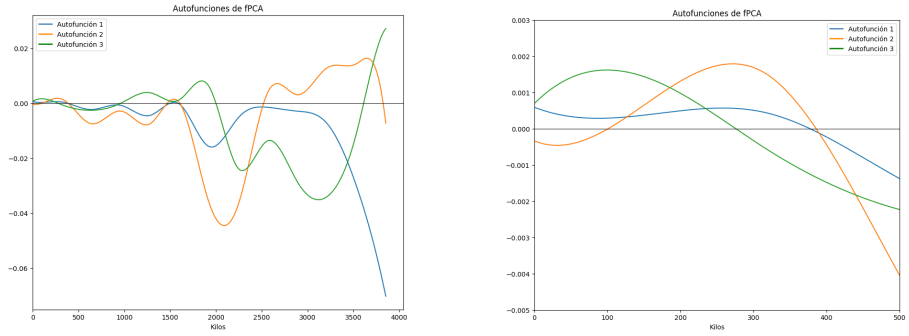
Observando los parámetros que componen a las mixturas en la Figura 4.4, vemos que a grandes rasgos las proyecciones están en torno a los valores que se esperaban. Lo que hicimos fue comparar los parámetros que predecimos para 2022 con los que se obtienen ajustando una mixtura para los lanzamientos de ese año. A excepción de los desvíos, los parámetros proyectados parecen estar en torno a los valores esperados, por lo que el desajuste respecto a la densidad original se debería a que la elección de dos lognormales no es la mejor para estos datos.

4.2. Método No Paramétrico

Para el método de PCA funcional, partimos de las funciones de densidad estimadas con kernels (o más bien, las log-densidades, de acuerdo a lo que se discutió en la Sección 2.2). Una diferencia respecto a las simulaciones es que, en este caso, re-expresamos las log-densidades en una base de 15 BSplines en lugar de 30. Luego, las proyectamos según sus tres primeros componentes principales, explicando entre los tres más de un 99 % de la variabilidad. De esta forma, pasamos de un problema de series de tiempo de funciones a una

serie de tiempo tradicional formada por las ponderaciones de las autofunciones. Ajustamos un modelo ARIMA para cada una de las tres series de *scores* y, con las predicciones, reconstruimos las densidades $\hat{f}_{T+h}$, tomando primero la exponencial y dividiendo por la integral para asegurarnos que sea una función de densidad.

Con la densidad reconstruida para 2022, se obtiene un KL de 0,06. En la Figura 4.7b se aprecia una representación bastante cercana a la densidad real, acertando en el pico en torno a los 200 kilos.



(a) Todo el dominio

(b) De 0 a 500 kilos

Figura 4.5: Inspección de las autofunciones. Haciendo foco en la región por debajo de los 500 kilos, la segunda autofunción parecería representar el pico de Starlink en torno a los 200-300 kilos, mientras que la tercera le da mayor peso a los satélites más livianos en torno al 0. La primera autofunción pondera positivamente toda la región, volviéndose negativa a partir de los 400 kilos.

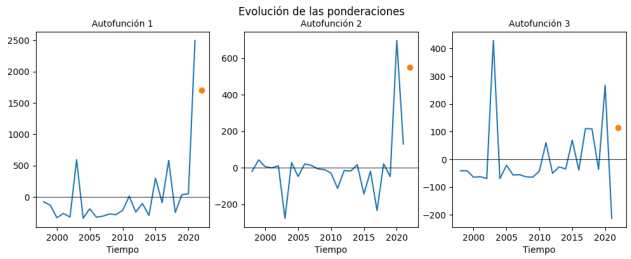


Figura 4.6: Evolución de las ponderaciones. Es interesante el comportamiento del *score* asociado a la primera autofunción. En la Figura 4.5b se ve que la autofunción toma valores positivos por debajo de los 400 kilos y valores negativos para masas superiores. Observando la serie de tiempo, notamos que en general esta función fue ponderada negativamente para casi todos los años de la muestra, para volverse positiva en los últimos años. Esto reflejaría el marcado salto en popularidad que se dio recientemente en los satélites pequeños en detrimento de los más pesados.

Analizando las autofunciones (Figura 4.5) y sus ponderaciones (Figura 4.6), vemos que para 2022 se les da una ponderación positiva a las tres, dándole mayor peso a la primera y menor peso a la última. Visualmente, la primera autofunción "castiga" fuertemente a las observaciones por encima de los 3000

kilos. Haciendo foco en la región entre los 0 y 500 kilos, vemos en la Figura 4.5b que la segunda autofunción tiene un pico entre los 200 y 300 kilos que no es contrarrestado por las otras (la primera autofunción toma valores positivos también en esta región, y la tercera se anula con un valor cercano a cero). Vemos en la Figura 4.6 que la segunda autofunción tiene una fuerte ponderación positiva, por lo que parecería ser la responsable de capturar el pico de Starlink.

Mirando los resultados en la Figura 4.7b, la función predicha también captura el pico en torno a los 0 kilos. Volviendo a la Figura 4.5b, en dicha región, tanto la primera como la tercera autofunción toman valores positivos y son ponderadas positivamente, por lo que serían las responsables de representar al grupo de satélites de menor peso. Por otro lado, su efecto se ve amortiguado, en parte, por la segunda autofunción que toma valores negativos, haciendo que este pico tenga menor relevancia en la densidad final que el pico en torno a los 200 kilos.

4.3. GANs Condicionales

Por último, para el método que se basa en GANs condicionales, ajustamos el modelo utilizando los mismos hiperparámetros que usamos previamente en las simulaciones. Algo a tener en cuenta, es que no ajustamos el modelo directamente sobre los datos, sino que, como paso previo, los reescalamos para que varíen entre 0 y 1 al igual que el vector de tiempo. Decidimos entrenar, al igual que con las simulaciones, durante 2000 iteraciones, si bien pasada la mitad del entrenamiento los resultados no registraron mejoras adicionales.

Una vez entrenado el modelo, generamos nuevas muestras condicionales al período que se desee, en nuestro caso para 2022, y en base a estas muestras ajustamos una función de densidad con kernels para contrastarla con la que se ajustó en base a los datos reales.

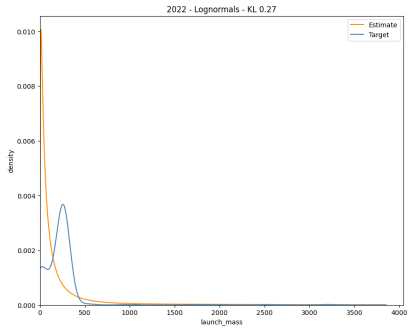
La densidad que termina resultando tiene una divergencia KL de 0,05 respecto a la densidad de test. En la Figura 4.7c vemos que el método logra acertar al pico de los 200 kilos con mayor precisión que el procedimiento de FPCA, si bien parece subestimar, en parte, la región en torno al 0.

4.4. Resultados

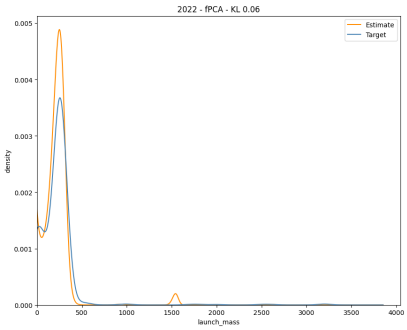
En la Figura 4.7 presentamos las densidades generadas por cada alternativa y las contrastamos con la densidad ajustada a los lanzamientos de 2022. El método paramétrico basado en una mixtura de dos lognormales genera el peor de los tres ajustes con una divergencia de Kullback-Leibler de 0,27. Si bien acierta en concentrar la densidad por debajo de los 500 kilos, no logra dar con la forma de la distribución real, pasando por alto la principal moda de la distribución (200 kilos) y concentrándose en cambio en torno al cero.

En segundo lugar en cuanto a desempeño, el procedmiento de FPCA logra un ajuste más fiel con un KL de 0,06, reconstruyendo la densidad como una combinación de las tres primeras autofunciones. No sólo acierta al pico de los 200 kilos, sino que también le da peso a la región en torno al 0. De todas formas, la densidad que asigna al pico de los 200 kilos es mayor de la que debería, y también da lugar a una ligera concentración en los 1500 kilos no observada en la densidad real.

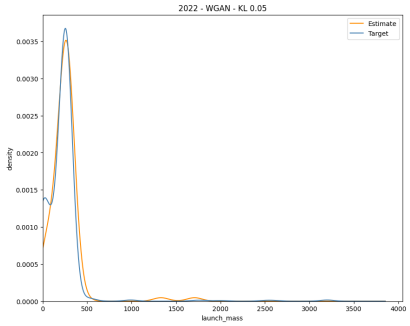
Por último, el mejor ajuste fue el generado por el modelo de redes adversarias, aunque se encuentra muy parejo con la alternativa de FPCA, con un KL de 0,05. La GAN identifica y le da el peso justo al pico de los 200 kilos, si bien se notan algunos defectos, como una menor densidad que la debida a la región en torno al 0. Recalcamos que los hiperparámetros y arquitecturas, tanto del generador como del discriminador, son los mismos que utilizamos en todo el trabajo, algo que se presenta como una ventaja del método, dando una noción de su versatilidad.



(a) Método Paramétrico



(b) Método No Paramétrico - FPCA



(c) GANs Condicionales

Figura 4.7: Predicciones 2022. El método de redes logra la mejor predicción obteniendo la menor divergencia Kullback-Leibler respecto a la densidad ajustada sobre los lanzamientos de ese año, acertando principalmente a la moda en torno a los 200 kilos.

CAPÍTULO 5

Conclusión

Se presentaron tres métodos alternativos para abordar el problema de la proyección de series de tiempo de funciones de densidad, mostrando su desarrollo teórico y su desempeño en sets de datos simulados y reales.

En primer lugar, el método paramétrico es una alternativa simple y relativamente rápida de implementar, que puede servir de *benchmark* para contrastar con otros métodos. Como vimos, puede tener un desempeño muy bueno si el tipo de funciones que se decide ajustar coincide con las funciones reales. Sin embargo, sufre ciertas desventajas, en especial a la hora de querer ajustar funciones más complejas que requieren más parámetros y que a su vez se traducen en más series de tiempo. Algo que no exploramos en este trabajo y que se presentaría como una ventaja de este método frente a los otros, es que puede ser útil si se quieren modelar datos generados por una familia de distribuciones discreta, como una Poisson o binomial; no habría que hacer ningún cambio para adaptar el método a este problema, ya que los pasos a seguir serían los mismos: ajustar los parámetros para el período de muestra y proyectarlos a futuro.

La segunda alternativa que presentamos se basa en Análisis de Componentes Principales Funcionales y es de las más populares en la literatura. Para el caso de aplicación, si bien quedó en segundo lugar en cuanto a desempeño, logró un ajuste similar al modelo de redes, aunque es cierto que requiere especial cuidado a la hora de interpretar y predecir las autofunciones y series de *scores*. Si bien es un método rápido de implementar, requiere que se le dedique cierto tiempo y algo de trabajo artesanal a la proyección de las series, ya que un mínimo cambio en ellas puede hacer que los resultados empeoren considerablemente. Además, si bien no es un método paramétrico, requiere la proyección de una cantidad de series de tiempo que puede escalar según la complejidad de las funciones; esto podría llegar a presentar un problema, por ejemplo, si se quieren proyectar funciones multivariadas. A su vez, vimos que el método presenta dificultades al querer proyectar una serie de densidades no estacionaria, como las que tratamos en las Secciones 3.1 y 3.2, donde las funciones no se encontraban definidas en un dominio acotado.

Por último, la alternativa de redes adversarias generativas logró el mejor desempeño de las tres en el caso de aplicación y en algunos de los sets simulados. El uso de este procedimiento para el pronóstico de funciones es innovador y no se encuentra mucha literatura sobre el mismo aplicado a este caso de uso puntual. Si bien de los tres métodos es el que más tarda en ajustarse, sigue siendo relativamente fácil de implementar; obtuvimos resultados satisfactorios, tanto en los datos simulados como en los reales, sin desviarnos prácticamente

de las arquitecturas utilizadas en otras implementaciones. En todas las corridas que hicimos en este trabajo entrenamos exactamente la misma GAN, utilizando el mismo conjunto de hiperparámetros y las mismas arquitecturas, tanto para el generador como el discriminador, por lo que parecería ser un método versátil. A diferencia de lo que plantean Zaheer et al. [13], no encontramos problemas utilizando este método para aprender distribuciones univariadas. Por el contrario, en algunos casos hasta logró un mejor ajuste y predicción a futuro que los otros métodos. Además, y como punto a explorar en futuros trabajos, debería ser sencillo escalar el método para aprender distribuciones más complejas y en más dimensiones, algo que podría resultar difícil con los otros métodos. De todas formas, uno de sus defectos es común a los modelos de redes neuronales: no deja de ser una caja negra; a diferencia de la alternativa de FPCA en la cual podíamos, por ejemplo, interpretar cada autofunción, este tercer método no ofrece mucho en cuanto a interpretabilidad.

En la Tabla 5.1 presentamos, a modo de resumen, una comparación entre las tres alternativas, mostrando sus principales ventajas y desventajas:

Método	**Paramétrico**	**FPCA**	**Redes Adversarias**
Ventajas	Simple y rápido de entrenar.	Buen desempeño en series estacionarias.	Buen desempeño aún sin cambiar hiperparámetros.
	Apto para distribuciones discretas.	Interpretabilidad de las autofunciones.	No requiere estimar series de tiempo ni densidades.
	No limitado a series estacionarias.		Escalable a dimensiones mayores.
Desventajas	Poca flexibilidad.	Depende del supuesto de estacionariedad.	Tiempo de entrenamiento.
	Poca escalabilidad a funciones más complejas y dimensiones más grandes.	Poca escalabilidad a mayores dimensiones.	Poco control e interpretabilidad sobre los resultados.
	Necesidad de modelar series de tiempo.	Necesidad de modelar series de tiempo.	

Tabla 5.1: Comparación de alternativas.

Como puntos a explorar a futuro podemos mencionar la aplicación de los métodos para proyectar la evolución de densidades multivariadas. De las tres alternativas que desarrollamos, la mejor candidata para distribuciones en dimensiones mayores es la de redes adversarias, teniendo en cuenta que fueron pensadas originalmente para generar datos complejos.

Por otro lado, se podría ampliar el análisis que hicimos en una sola variable, explorando también la estabilidad de los tres métodos. Tanto en los datos simulados como en el caso de aplicación, nos enfocamos en la calidad del ajuste que hacían los métodos, pero no analizamos la variabilidad de los mismos.

Hyndman y Shang cubren este tópico proponiendo utilizar métodos de *bootstrap* para construir intervalos de confianza para las funciones estimadas, si bien no lo hacen con funciones de densidad [6].

Bibliografía

[1] Arjovsky, M., Chintala, S. and Bottou, L. (2017). "Wasserstein GAN," arXiv preprint arXiv:1701.07875.

[2] Fabbri, C. (2017). "Conditional Wasserstein Generative Adversarial Networks," https://cameronfabbri.github.io/papers/conditionalWGAN.pdf

[3] Goodfellow, I., Pouget-Abadie J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio Y. (2014). "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672–2680.

[4] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. and Courville, A. (2017). "Improved Training of Wasserstein GANs," in NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, December 2017, Pages 5769–5779.

[5] Horta, E. and Ziegelmann, F. (2018). Dynamics of financial returns densities: A functional approach applied to the Bovespa intraday index. International Journal of Forecasting 34 (2018) 75–88.

[6] Hyndman, R.J. and Shang, H.L. (2009). "Forecasting functional time series," in Journal of the Korean Statistical Society 38 (2009) 199-211.

[7] Kullback, S. and Leibler, R.A. (1951). "On information and sufficiency". Annals of Mathematical Statistics. 22 (1): 79–86. doi:10.1214/aoms/1177729694.

[8] Mirza, M. and Osindero, S. (2014). "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784.

[9] Nicol, F. (2013). Functional Principal Component Analysis of Aircraft Trajectories. [Research Report] RR/ENAC/2013/02, ENAC. hal-01349113.

[10] Ramsay, J.O. and Silverman, B.W. (2005). Functional Data Analysis. Springer, New York. http://dx.doi.org/10.1002/0470013192.bsa239.

[11] Sen, R. and Ma, C. (2015). Forecasting Density Function: Application in Finance. Journal of Mathematical Finance, 5, 433-447. doi: 10.4236/jmf.2015.55037.

[12] Silverman B.W. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall, London.

[13] Zaheer, M., Li, C., Póczos, B. and Salakhutdinov, R. (2017). "GAN Connoisseur: Can GANs Learn Simple 1D Parametric Distributions?," In NIPS Workshop on Deep Learning: Bridging Theory and Practice.