Desmitificando Facebook Prophet:

Fundamentos y Aplicación Práctica al Análisis de Series Temporales

Análisis Técnico por: Arian Pedroza, Miguel Cuéllar, Edgar Godinez

Junio de 2025

Resumen

Abstract. El modelo Prophet, desarrollado por Facebook, se ha convertido en una herramienta estándar para el pronóstico de series temporales debido a su accesibilidad y robustez. Sin embargo, su facilidad de uso a menudo oculta la sofisticada maquinaria estadística que opera internamente. Este documento técnico tiene como objetivo desmitificar el funcionamiento de Prophet, proporcionando una guía detallada que va desde la intuición conceptual hasta la formulación matemática rigurosa. Se analizan en profundidad sus componentes clave: la tendencia mediante regresión por tramos, la estacionalidad a través de Series de Fourier, y el rol crucial de la regularización L1 y L2. Además, se explora el marco de inferencia Bayesiana y los algoritmos de optimización que sustentan el proceso de ajuste del modelo. El propósito de este texto es dotar a estudiantes y profesionales de datos de una comprensión estructural que les permita utilizar Prophet no como una caja negra, sino como una herramienta potente y controlada, justificando decisiones metodológicas clave como la transformación de datos y el ajuste de hiperparámetros.

Índice

1. Fundamentos: El Preprocesamiento como Pilar del Modelo

La calidad de un modelo predictivo está intrínsecamente ligada a la calidad de los datos con los que se entrena. Al analizar datos de conteo, como el número de robos diarios (y_{orig}), es imperativo abordar sus características estadísticas antes del modelado.

1.1. Diagnóstico: La Distribución de los Datos de Conteo

Al visualizar la distribución de frecuencias de los robos diarios, se observa un patrón típico de los **datos de conteo**: un fuerte **sesgo positivo**. La mayoría de las observaciones (más del 70 %) se concentran en valores bajos (0-5 robos), mientras que una çola"larga se extiende hacia la derecha, representando los pocos días con una incidencia excepcionalmente alta. Ignorar esta estructura tiene dos consecuencias perjudiciales.

1.1.1. Problema 1: Influencia Desproporcionada de Outliers

Los modelos de regresión, como los que Prophet utiliza, se optimizan minimizando una **función de pérdida**. La más común es el **Error Cuadrático Medio (MSE)**, $\text{MSE} = \frac{1}{n}\sum(y_{\text{real}} - y_{\text{predicho}})^2$. El exponente al cuadrado es la fuente del problema.

Explicación Técnica: La Tiranía del Error al Cuadrado. La penalización por un error crece cuadráticamente, lo que da un peso desproporcionado a los errores grandes.

- **Día con incidencia baja:** Si ocurren 3 robos y el modelo predice 5, el error es -2. La contribución a la pérdida total es $(-2)^2 = \mathbf{4}$.
- **Día con incidencia alta (outlier):** Si ocurren 25 robos y el modelo predice 5, el error es 20. La contribución a la pérdida total es $(20)^2 = \mathbf{400}$.

El modelo, en su esfuerzo por minimizar la pérdida total, dedicará una cantidad desproporcionada de sus recursos a reducir el error de 400, incluso si eso implica un peor ajuste para la mayoría de los días normales.

1.1.2. Problema 2: Violación de Supuestos (Heterocedasticidad)

El segundo problema es la **heterocedasticidad** ("diferente dispersión"), que viola el supuesto de que la varianza de los errores es constante (**homocedasticidad**).

Explicación Técnica: Varianza Dependiente de la Media.

- **Cuando la media de robos es baja (ej. 2):** El rango de resultados está naturalmente limitado por el suelo en cero. Es probable que los valores observados se muevan en un rango estrecho (ej. 0 a 4), resultando en una varianza pequeña.

- **Cuando la media de robos es alta (ej. 25):** El límite en cero ya no es una restricción efectiva. El rango de resultados plausibles se amplía significativamente (ej. 15 a 35), lo que resulta en una varianza mucho mayor.

Esta variabilidad no constante invalida los **intervalos de confianza**, que se vuelven demasiado anchos para predicciones bajas y peligrosamente estrechos para predicciones altas.

1.2. Solución: La Transformación Logarítmica

La transformación $y_{\text{transf}} = \log(1 + y_{\text{orig}})$ es el tratamiento preciso para estos problemas.

1.2.1. Efecto 1: Domando a los Outliers (Compresión de Escala)

Ejemplo Numérico: El Efecto de Compresión del Logaritmo

Usemos el logaritmo natural (ln) para cuantificar cómo la transformación altera la "percepción"del modelo sobre los cambios en los datos.

- **Impacto en el Rango Común (1 a 5 robos):** En la escala original, un aumento de 4 unidades. En la escala logarítmica, el cambio percibido por el modelo es:
$$\ln(1+5) - \ln(1+1) \approx 1{,}792 - 0{,}693 = \mathbf{1{,}099}$$
- **Impacto en el Rango Atípico (20 a 24 robos):** En la escala original, es el mismo aumento de 4 unidades. Sin embargo, en la escala logarítmica, el cambio es:
$$\ln(1+24) - \ln(1+20) \approx 3{,}219 - 3{,}045 = \mathbf{0{,}174}$$

Conclusión del Cálculo: El impacto que un cambio de 4 robos tiene en el modelo es drásticamente diferente dependiendo de la zona de la distribución. Al dividir ambos impactos ($1{,}099/0{,}174 \approx 6{,}3$), vemos que un cambio en el rango alto tiene **más de 6 veces menos influencia** que un cambio idéntico en el rango bajo. De esta manera, el logaritmo "doma.ª los outliers, evitando que su gran magnitud distorsione el ajuste general del modelo.

1.2.2. Efecto 2: Estabilizando la Varianza

Al comprimir los valores altos, la transformación acerca la serie al supuesto de homocedasticidad, permitiendo el cálculo de intervalos de confianza fiables.

2. La Anatomía de Prophet: El Encendido del Motor

La arquitectura de Prophet se basa en la descomposición de la serie temporal en componentes interpretables. Pero, ¿cómo pasa el modelo de un simple DataFrame de dos columnas (‘ds‘, ‘y‘) a un pronóstico sofisticado? Vamos a seguir el proceso paso a paso.

La ecuación maestra es:

$$y(t) = g(t) + s(t) + h(t) + r(t) + \epsilon(t) \tag{1}$$

Donde $y(t)$ es nuestra serie transformada $\log(1 + \text{robos})$. El objetivo del .ªjuste"(‘model.fit()‘) es encontrar los mejores parámetros para cada uno de estos componentes.

Analogía: La Partitura Musical

Piense en la serie temporal $y(t)$ como la melodía final que queremos tocar. Prophet primero escribe una **partitura musical** (la matriz de diseño), donde cada columna es la parte de un instrumento (un violín para la tendencia, una flauta para la estacionalidad, etc.). El proceso de ‘fit()‘ es como encontrar los "volúmenes"(los coeficientes) correctos para cada instrumento para que, cuando toquen todos juntos, su suma recree la melodía original.

2.1. Una Pausa Necesaria: Entendiendo la Regularización L1 y L2

Antes de ver cómo se construyen los componentes, es crucial entender la herramienta principal que usa Prophet para evitar el sobreajuste: la **regularización**.

Concepto Clave: ¿Qué es la Regularización?

Es una técnica que añade una "penalización.ª la complejidad del modelo. Obliga al modelo a justificar cada parámetro flexible: solo se permite si reduce el error de predicción más de lo que cuesta la penalización. Esto fomenta modelos más simples y robustos. Prophet usa dos tipos: **L1** para la tendencia y **L2** para la estacionalidad.

2.1.1. Regularización L1 (Lasso): El Selector de Características

- **Penalización:** Proporcional a la **suma de los valores absolutos** de los coeficientes: $\lambda \sum |\beta_j|$.
- **Efecto (Sparsity):** Su propiedad matemática única es que puede forzar a que algunos coeficientes se vuelvan **exactamente cero**. Por lo tanto, realiza una **selección de características** automática, eliminando las que no son importantes.

- **Analogía (El Equipo de Estrellas):** L1 actúa como un entrenador que solo quiere superestrellas. Si un jugador (un changepoint) no tiene un impacto masivo, lo envía a la banca (su coeficiente se hace cero). El resultado es un equipo titular pequeño pero potente.
- **Uso en Prophet:** Perfecto para la tendencia. Queremos asumir que es estable y solo permitir cambios (activar un δ_j) donde la evidencia es abrumadora.

2.1.2. Regularización L2 (Ridge): El Suavizador Colaborativo

- **Penalización:** Proporcional a la **suma de los cuadrados** de los coeficientes: $\lambda \sum \beta_j^2$.
- **Efecto (Shrinkage):** Penalizar el cuadrado hace que sea muy çostoso"tener coeficientes grandes. Como resultado, L2 tiende a **encoger (shrink)** todos los coeficientes hacia cero de manera uniforme, pero rara vez los hace exactamente cero.
- **Analogía (El Equipo Colaborativo):** L2 actúa como un entrenador que valora el trabajo en equipo. En lugar de eliminar jugadores, les pide a todos que moderen su juego y contribuyan un poco, resultando en un juego colectivo fluido y suave.
- **Uso en Prophet:** Perfecto para la estacionalidad. Queremos una mezcla suave de muchas ondas que trabajen juntas. L2 logra esto encogiendo las amplitudes de todas las ondas de manera coordinada.

2.2. El Motor Principal $g(t)$: Construyendo la Tendencia

2.2.1. Paso 1: Poner los Postes (Changepoints Candidatos)

El primer paso es mecánico. Prophet coloca **changepoints candidatos** (por defecto 25) de forma **equidistante** a lo largo del primer 80 % de los datos. Son solo "posibilidades".

2.2.2. Paso 2: Escribir la Partitura de la Tendencia (La Matriz A)

Para cada fecha t, Prophet crea una fila de características para la tendencia. Si hay S changepoints candidatos, esta parte de la matriz tendrá $S + 1$ columnas:

- **Columna 1 (Base):** El tiempo, t (ej. 0, 1, 2...).
- **Columnas 2 a S+1 (Interruptores):** El vector $\mathbf{a}(t)$. Para cada changepoint s_j, hay una columna que contiene un 0 para todas las fechas anteriores a s_j y un 1 para todas las posteriores.

2.2.3. Paso 3: Definir los Parámetros y Ajustar con L1

Prophet define los parámetros a aprender: la pendiente base k, el intercepto m, y el vector de ajustes de pendiente $\boldsymbol{\delta}$. Luego, el optimizador busca los valores que minimizan el error + la

penalización L1:

$$\min_{k,m,\boldsymbol{\delta}} \left(\sum (\text{error})^2 + \lambda \sum_{j=1}^{S} |\delta_j| \right)$$

El resultado es un vector $\boldsymbol{\delta}$ **disperso** (lleno de ceros), donde solo los changepoints más importantes tienen un $\delta_j \neq 0$.

2.2.4. Paso 4: Ensamblaje Final de $g(t)$

Con los parámetros ya encontrados, Prophet calcula la tendencia para cualquier fecha t usando la fórmula completa, que ahora podemos entender en su totalidad:

$$g(t) = \underbrace{(k + \mathbf{a}(t)^T \boldsymbol{\delta})}_{\text{Pendiente en el tiempo } t} \cdot t + \underbrace{(m + \mathbf{a}(t)^T \boldsymbol{\gamma})}_{\text{Intercepto en el tiempo } t}$$

Donde $\gamma_j = -s_j \delta_j$ es un ajuste calculado para asegurar la continuidad de la línea.

2.3. La Sección Rítmica $s(t)$: Construyendo la Estacionalidad

2.3.1. Paso 1: Escribir la Partitura Estacional (Matriz $\mathbf{X}_s$)

Para cada tipo de estacionalidad (ej. semanal), Prophet genera columnas en la matriz de diseño usando pares de funciones seno y coseno de diferentes frecuencias. Para una estacionalidad semanal con $N = 3$, generará 6 columnas: $\cos(2\pi t/7), \sin(2\pi t/7), \cos(4\pi t/7), \ldots$

2.3.2. Paso 2: Definir Parámetros y Ajustar con L2

El modelo necesita aprender los coeficientes $\boldsymbol{\beta} = [a_1, b_1, \ldots, a_N, b_N]^T$ para estas columnas. El optimizador minimiza el error + la penalización L2:

$$\min_{\boldsymbol{\beta}} \left(\sum (\text{error})^2 + \lambda \sum_{j=1}^{2N} \beta_j^2 \right)$$

Esto .^encoge"los coeficientes, resultando en un patrón estacional suave.

2.3.3. Paso 3: Ensamblaje Final de $s(t)$

Una vez encontrado $\boldsymbol{\beta}$, la estacionalidad es un producto punto: $s(t) = \mathbf{X}_s(t) \cdot \boldsymbol{\beta}$.

2.4. El Ensamblaje Final del Modelo Completo

Prophet no ajusta cada componente por separado. Construye una **única y gigantesca matriz de diseño** con todas las columnas (tendencia, estacionalidad, festivos, regresores). Luego, resuelve **un único gran problema de regresión** para encontrar todos los parámetros simultáneamente, aplicando la regularización L1 a los δ de la tendencia y L2 a todos los demás coeficientes.

3. El Motor de Inferencia: ¿Cómo .ªprende"Prophet?

3.1. Marco Bayesiano y Estimación Máximo a Posteriori (MAP)

El Teorema de Bayes nos da un marco para combinar nuestras creencias previas con la evidencia de los datos:

$$\underbrace{P(\theta|D)}_{\text{Posterior}} \propto \underbrace{P(D|\theta)}_{\text{Verosimilitud}} \times \underbrace{P(\theta)}_{\text{Prior}}$$

Prophet, por defecto, encuentra la **Estimación Máximo a Posteriori (MAP)**: el punto más alto (el modo) de la distribución posterior.

3.2. El Motor Computacional: Stan y el Optimizador L-BFGS

Prophet utiliza **Stan**, una plataforma de programación probabilística, para realizar la optimización. Stan emplea por defecto el algoritmo **L-BFGS**.

Analogía: L-BFGS, el Excursionista Inteligente en la Niebla

Imagina que buscas la cima de una montaña (el punto MAP) en una noche con niebla.

- **Descenso de Gradiente (simple):** Solo sientes la pendiente (gradiente) y das un paso.
- **L-BFGS (cuasi-Newton):** No solo usa la pendiente, sino que recuerda"la información de sus últimos movimientos para construir una aproximación de la **curvatura** del terreno. Esto le permite dar pasos más largos y directos. La "L"de "Limited-memory"lo hace extremadamente eficiente.

4. Conclusión

Facebook Prophet representa un avance significativo en el pronóstico de series temporales, no solo por su precisión, sino por su diseño deliberado en torno a la **interpretabilidad y la facilidad de ajuste**. Su arquitectura de descomposición aditiva, que separa la tendencia, la estacionalidad y los efectos de eventos, permite a los analistas diagnosticar el comportamiento de una serie y cuantificar el impacto de diversos factores.

La verdadera sofisticación de Prophet reside en su uso pragmático de técnicas estadísticas avanzadas. La regularización Laplaciana (L1) para la selección de changepoints y la regularización Gaussiana (L2) para suavizar la estacionalidad son ejemplos de cómo el modelo equilibra la flexibilidad con la robustez para evitar el sobreajuste. Todo este marco se implementa de manera eficiente a través de la inferencia bayesiana y motores de optimización de última generación como Stan y L-BFGS.

Comprender estos mecanismos internos es fundamental. Transforma al usuario de un mero implementador a un analista crítico, capaz de tomar decisiones informadas sobre la transformación

de datos, el ajuste de hiperparámetros y la interpretación de los resultados. Este conocimiento no solo conduce a mejores modelos, sino también a una mayor confianza en las conclusiones y recomendaciones que se derivan de ellos.