Pasar de un modelo tan estructurado a su implementación computacional no resulto fácil. Sin embargo, se logró desarrollar un algoritmo que estima todos los parámetros del modelo de una forma eficiente y que funciona en la práctica. En el fondo, el algoritmo recae en el método de Gibbs sampling propuesto en (**albert1993bayesian**), por lo que se hace una breve introducción a la escuela de inferencia bayesiana, y en el algortimo de backfitting descrito en (**hastie1986generalized**). Al algoritmo se le titula: *bayesian piece wise polinomial model (bpwpm)*. Para facilitar la utilización del modelo en diversas bases de datos, así como su validación y visualización, a la par del algoritmo se desarrolló un paquete de código abierto (con el mismo nombre) para el software estadístico `R`. A darle un tratamiento bayesiano a los parámetros, más que estimarlos, se busca regresar una muestra de tamaño arbitrario de sus correspondientes distribuciones posteriores. La idea, es que estas distribuciones posteriores, se haya capturado toda la información de los datos de entrenamiento.

Se considera, que una buena forma de entender el algoritmo es *visualizando* tanto los datos como los objetos que componen el modelo, por lo tanto se hace un paréntesis notacional. De las ecuaciones del modelo: **??** a **??**, se tienen dos grupos de parámetros por estimar, $\beta \in \mathbb{R}^{d+1}$ y $w_j \in \mathbb{R}^{N^*} \quad \forall j = 1, \ldots, d$. Donde:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \quad \text{y} \quad w_j = \begin{bmatrix} w_1 \\ \vdots \\ w_{N^*} \end{bmatrix}$$

Se hace énfasis en que existen d vectores w_j, cada uno de tamaño N^*. Por lo tanto, se tienen sun total de $1 + d + dN^*$ parámetros. Se usa el símbolo $\mathbf{w}$ para designar todos los vectores w_j, haciendo de este una matriz, es decir: $\mathbf{w} \in \mathbb{R}^{d \times N^*}$. Cuando se habla de datos: $\{(y_i, \mathbf{x}_i)\}_{i=1}^n$, estos se pueden representar en una tabla (o matriz):

$$\left[\begin{array}{c|ccc} y_1 & x_{1,1} & \ldots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ y_n & x_{n,d} & \ldots & x_{n,d} \end{array}\right]$$

Donde el vector de observaciones binarias $\mathbf{y} = (y_1, \ldots, y_n)^t$ es la primer columna de la tabla, y la matriz de covariables $\mathbf{X}$ es el resto.

Bajo esta representación, se da contexto cuando se habla de que la estimación debe reflejar los patrones *hacia abajo* y *hacia lo largo*. Hacia abajo, se está captando la información existente entre las observaciones; cada f_j, mediante su parámetro w_j, representa una transformación no lineal de la variable (o dimensión) j.

Hacia lo largo, la función de proyección f suma cada f_j a través de β, ponderando los efectos individuales de cada variable. Mantener el balance entre la estimación de β a lo largo y w_j hacia abajo, es fundamental para el algoritmo. Analizando este hecho, se concluye que la estimación de ambos grupos de parámetros, se puede ver como una regresión separada para cada uno, y por ende, estos pueden ser estimados por el mismo algoritmo. Esto responde a la dualidad que se exploró en el capítulo pasado de que ambas expresiones son expansiones en bases funcionales. El puente que conecta, y controla el balance entre ambas, son los residuales parciales. Los siguientes capítulos, se concentran en explicar e implementar este curioso patrón.

0.1. Fundamentos de la estadística bayesiana

- Hacer preambulo bayesiano y justificación de la fislosofía bayesiana - Explicar formula de bayes y actualización bayesiana

0.1.1. Funciones de probabilidad condicional completas

- Mencionar que existen y que indúcen un Gibbs Sampler por pedazos

0.2. Simulación bayesiana: el Gibbs sampler

- Explicar que el Gibbs sampler es un método MCMC.

0.2.1. Algoritmo de Albert y Chibb

- Hacer derivación de la condicional para β paper de Albert + Chibb - Poner pseudocódigo - Argumentar por qué es igual para w pero en lugar de usar las z de regresores se usan los residuales parciales. - Mencionar que en el paper se hacen más algoritmos para diferentes cosas

0.2.2. Especificación probabilistica para el modelo

Para los parámetros, se usan las siguientes distribuciones *apriori*:

$$\beta = (\underline{e}ta_0, \beta_1, \ldots, \beta_d)^t \sim N_d(\mu_0, \Sigma_0) \tag{1}$$

$$w^{(i)} = (w_1^{(i)}, \ldots, w_J^{(i)})^t \sim N_J(\mu_0^{(i)}, \Sigma_0^{(i)}) \quad i = 1, \ldots, d \tag{2}$$

0.3. Algoritmo *bpwpm*

En forma de pseudocódigo el algoritmo tiene la siguiente forma:
A diferencia de la exposición del modelo, el algoritmo debe de construir de abajo hacia arriba, pues se necesita tener una estimación puntual de los parámetros para poder calcular las funciones intermedias y que todo quede definido de forma numérica.

```
Parametros inciales:
WHILE (...)
    Transformación de X -> Phi -> F (Función: estimate_PWP)
    Simulación de betas (Función simulate_beta)
```

- Hacer énfasis en el apéndice y el paquete.

0.3.1. Algoritmo de *backfitting* para ajuste de modelos GAM

- Justificación final para las w's
- Usamos los nodos iniciales en cuantiles determinados.

- taus: HMC
- β Estimar por máxima verosimilitud pero dentro del Gibbs con el método ABC
- $w's$ BAyesianas + importantes que las betas.

- Explicar Alortimo y hacer pseudocódigo de cada sección - Explicar lógica del algoritmo - Explicar desarrolo de paquetes en R - Explicar bien la parte de los residuales y el algorimto backfitting, por que las f_j son arbitrarias y pueden interpolar a los residuales para hacer el ajuste. Esto también explica las β pues si se pueden capturar chigón los residuales con una sola dimensión, te vale verga la siguiente :). Yei bitches

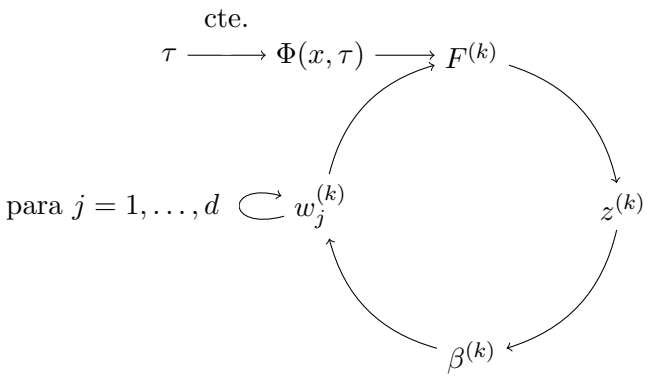

Figura 1: Esquema del algoritmo