LIBRERÍA EN C++ PARA PROCESOS GAUSSIANOS

Procesos Gaussianos para Regresión

Carlos González
Alejandro Suárez
Manuel Pineda

26 de febrero de 2016

Universidad Tecnológica de Pereira

INTRODUCCIÓN

REGRESIÓN

Los métodos de regresión buscan establecer o estimar las relaciones entre variables dependientes e independientes.

Son utilizados usualmente como métodos de predicción y su uso le hace coincidir frecuentemente con técnicas de Machine Learning.

PROCESO GAUSSIANO

Puede definirse como una distribución normal sobre funciones, y en términos generales se puede notar como una distribución normal multivariada.

$$\mathcal{GP} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{1}$$

PROCESOS GAUSSIANOS PARA REGRESIÓN

En general, busca explicar las relaciones entre las variables a partir de la distribución normal multivariada que lo representa.

Este método tiene grandes ventajas derivadas de las propiedades naturales a la distribución normal, si bien es costoso computacionalmente[1].

[1] Un proceso de regresión completo utilizando procesos Gaussianos tiene una complejidad de $O(n^3)$.

EL CONCEPTO DE LIBRERÍA

Es un conjunto de implementaciones que proveen distintos tipos de funcionalidad a través de una API bien definida, y pueden ser usadas por otros desarrolladores dentro de su propio código.

LIBRERÍAS EXISTENTES

Actualmente una de las librerías líder en el campo de los procesos Gaussianos se conoce como GPy (desarrollada en Python). Otras implementaciones existen en otros lenguajes orientados a la matemática, como MATLAB.

MOTIVACIÓN

El candidato a PhD Sebastián Gómez habría desarrollado anteriormente una implementación de los procesos para Regresión y Clasificación en MATLAB, él es quien motiva la creación de una nueva implementación, pero esta vez a manera de librería y en un lenguaje abierto y compilado, orientado al desempeño.

OBJETIVOS DEL PROYECTO

- Crear una librería en un lenguaje compilado y libre (C++) para procesos Gaussianos con una API amigable.
- Implementar procesos Gaussianos para Regresión.
- Implementar una de las aproximaciones para reducir la complejidad del proceso.
- Garantizar el correcto funcionamiento de la librería a partir del uso de un framework de pruebas.
- Verificar desempeño a través de medición de tiempos de ejecución usando la Regresión completa y la aproximación.

Implementación de la librería

ADMINISTRACIÓN

- Collective Code Construction Contract, C4.
- Pruebas automáticas utilizando BOOST.
- Integración continua usando Travis.
- Documentación generada a través de doxygen.

DEPENDENCIAS

- Armadillo, libería de algebra lineal en C++.
  - LAPACK.
  - openBLAS.
  - ARPACK.[2]
- NLopt, libería de optimización en C++.

[2] Estas dependencias permiten el uso del varios núcleos en procesador

INSTALACIÓN

- `make`
- `make check`[3]
- `make install`
- `make installcheck`[4]

[3] Opcional, Prueba el Build
[4] Opcional, Prueba la instalación

REGRESIÓN COMPLETA

Recordemos de (1).

$$\mathcal{GP} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Ahora bien en la regresión, si se tiene X_1 y X_2, conjunto de datos no observados y observados respectivamente:

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{2}$$

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \tag{3}$$

Distribución predictiva (condicionada)

Dada por:

$$\mathcal{N}(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}) \tag{4}$$

Donde

$$\bar{\mu} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y - \mu_2) \tag{5}$$

$$\bar{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{6}$$

COVARIANZA Y MEDIA

Ahora bien, se necesita calcular Σ y μ para obtener $\bar{\Sigma}$ y $\bar{\mu}$, a través de una función kernel:

$$\Sigma = K(X, X) \tag{7}$$

Con

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{8}$$

KERNEL LMC

El kernel de múltiple salida utilizado es el kernel LMC (Linear Model of Corregionalization):

$$K(X, Y) = \sum_{q=1}^{Q} \beta_q \otimes k_q(X, Y) \tag{9}$$

Donde cada k_q representa una función kernel de única salida, para nuestro caso el squared exponential:

$$k(x, y) = \sigma^2 e^{-\frac{(x-y)^2}{2l^2}} \tag{10}$$

CRITERIO DE OPTIMIZACIÓN

Si se desea una buena aproximación se deben optimizar los parámetros del kernel, la función a optimizar es el log likelihood:

$$\ln(L) = -\frac{1}{2}\ln(|\Sigma|) - \frac{1}{2}y^T\Sigma^{-1}y - \frac{n}{2}\ln(2\pi) \tag{11}$$

Y teniendo en cuenta el ruido (σ_{noise}):

$$\ln(L) = -\frac{1}{2}\ln(|\Sigma + \sigma^2_{noise}I|) - \frac{1}{2}y^T(\Sigma + \sigma^2_{noise}I)^{-1}y - \frac{n}{2}\ln(2\pi) \tag{12}$$

DERIVADA DEL LOG LIKELIHOOD

Considerando que se utilizará un optimizador basado en gradiente, se hace imprescindible derivar la función objetivo. Sea θ_i un parámetro arbitrario, y $R = \Sigma + \sigma^2_{noise} I$ la derivada viene dada por:

$$\frac{\partial}{\partial \theta_i} \ln(L) = -\frac{1}{2} tr(R^{-1} \frac{\partial R}{\partial \theta_i}) + \frac{1}{2} y^T R^{-1} \frac{\partial R}{\partial \theta_i} R^{-1} y \tag{13}$$

APROXIMACIÓN FITC

FULLY INDEPENDENT TRAINING CONDITIONAL

Usa el concepto de puntos ocultos (pseudo-inputs) para reducir la complejidad computacional del proceso[5], la distribución predictiva está dada, como en la regresión completa, por una normal multivariada $\mathcal{N}(\mu, \Sigma)$, donde:

$$\mu = Q_{*,f}(Q_{f,f} + \Lambda)^{-1} y \tag{14}$$

$$\Sigma = K_{*,*} - Q_{*,f}(Q_{f,f} + \Lambda)^{-1} Q_{f,*} \tag{15}$$

[5] La complejidad de FITC es $O(kn^2)$ donde k es el número de puntos ocultos

ECUACIONES ADICIONALES

Para las ecuaciones anteriores:

$$Q_{x,y} = K_{x,u} K_{u,u}^{-1} K_{u,y} \tag{16}$$

Donde u representa el conjunto de puntos ocultos. Adicionalmente Λ está dado por:

$$\Lambda = \text{diag}[K_{f,f} - Q_{f,f}] + \sigma^2_{\text{noise}} I \tag{17}$$

LOG LIKELIHOOD PARA FITC

Finalmente la función Log Likelihood para FITC sería:

$$\ln(L) = -\frac{1}{2}\ln(|Q_{f,f} + \Lambda|) - \frac{1}{2}y^T(Q_{f,f} + \Lambda)^{-1}y - \frac{n}{2}\ln(2\pi) \qquad (18)$$

DERIVADA DEL LOG LIKELIHOOD

Sea θ_i un parámetro arbitrario de la función objetivo y:

$$R = Q_{ff} + \text{diag}[K_{ff} - Q_{ff}] + \sigma^2_{\text{noise}} I \tag{19}$$

Entonces

$$\frac{\partial \log(y|X)}{\partial \theta_i} = -\frac{1}{2}\text{tr}\left(R^{-1}\frac{\partial R}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{y}^\top R^{-1}\frac{\partial R}{\partial \theta_i}R^{-1}\mathbf{y} \tag{20}$$

DERIVADAS ADICIONALES

De acuerdo con la definición de R:

$$\frac{\partial R}{\partial \theta_i} = \frac{\partial Q_{ff}}{\partial \theta_i} + \text{diag}[\frac{\partial K_{ff}}{\partial \theta_i}] - \text{diag}[\frac{\partial Q_{ff}}{\partial \theta_i}] + \frac{\partial \sigma^2_{noise} I}{\partial \theta_i} \tag{21}$$

y

$$\frac{\partial Q_{ff}}{\partial \theta_i} = K_{fu} K_{uu}^{-1} \Big[\frac{\partial K_{uf}}{\partial \theta_i} - (\frac{\partial K_{uu}}{\partial \theta_i} K_{uu}^{-1}) K_{uf}\Big] + \frac{\partial K_{fu}}{\partial \theta_i} K_{uu}^{-1} K_{uf} \tag{22}$$

FUNCIONALIDAD Y DESEMPEÑO

Para cada tipo de Proceso Gaussiano se realizaron pruebas de funcionalidad, haciendo regresión sobre la función sinc_N normalizada, definida como:

$$\text{sinc}_N(x) = \frac{\text{seno}(\pi x)}{\pi x} \tag{23}$$

Para generar ruido a las funciones se utilizó σ como variable aleatoria normalmente distribuida con media 0 y varianza 0,0001, para las pruebas se utilizaron 150 puntos y para optimizar máximo 1000 iteraciones con una tolerancia de $10e^{-4}$, además para la aproximación FITC se utilizaron 30 puntos ocultos.

PREDICCIÓN GP DE ÚNICA SALIDA

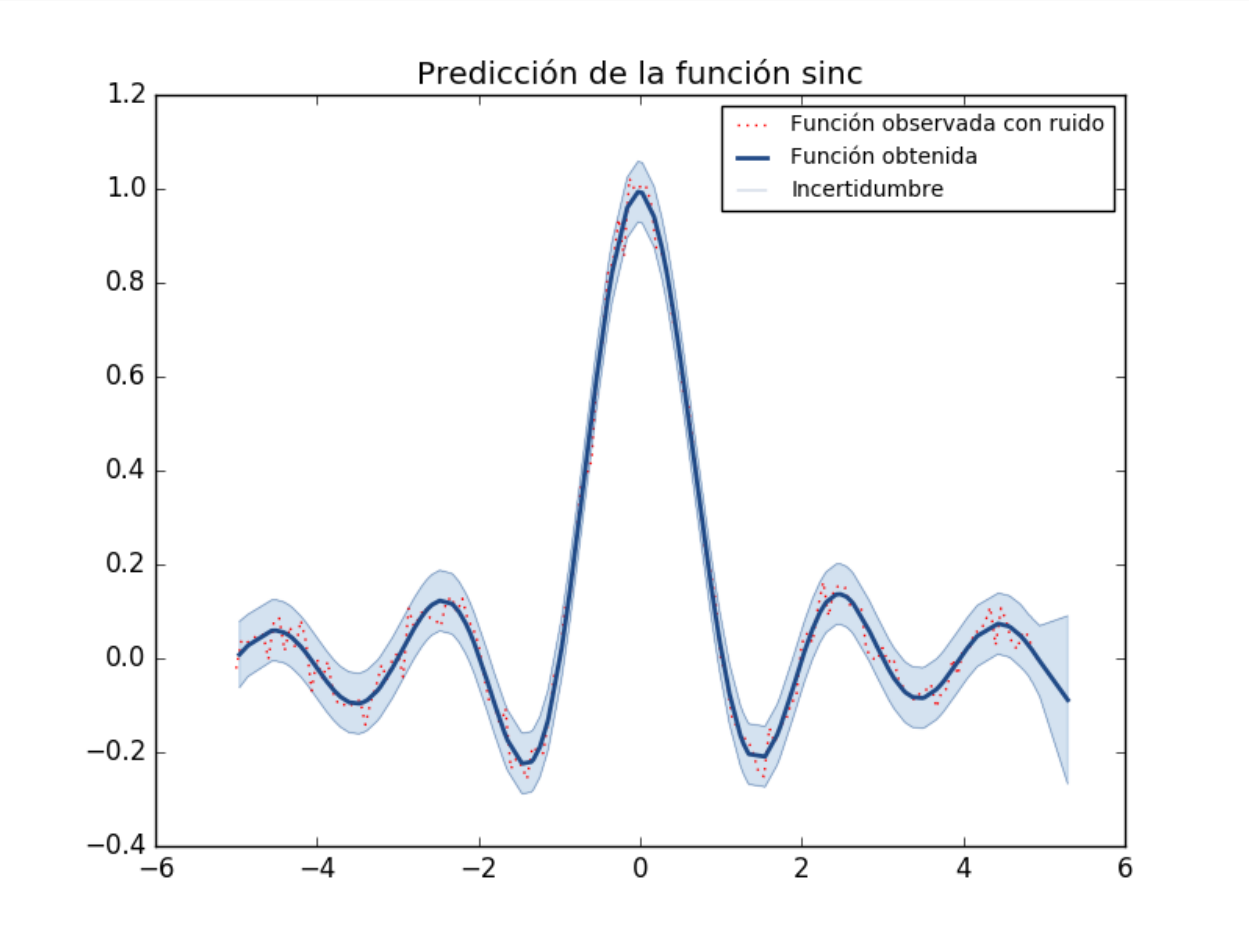
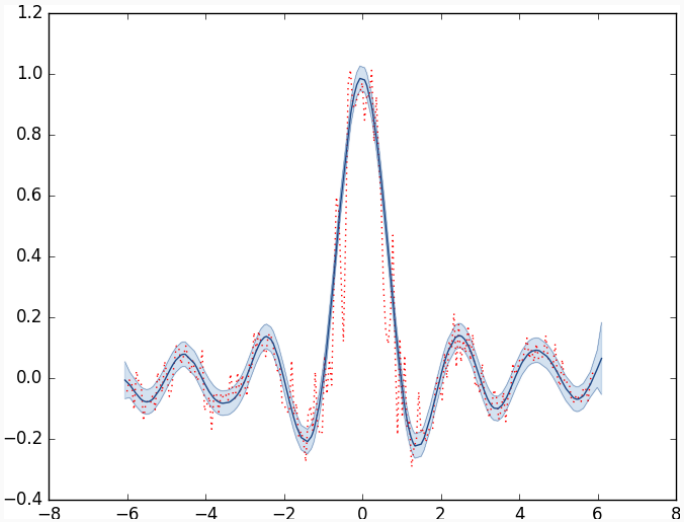
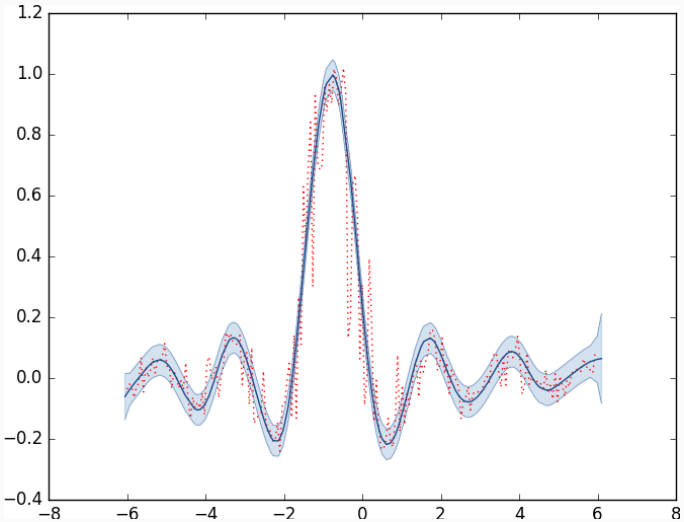


Figura: Predicción GP de única salida
$\text{sinc}(x) + \sigma$

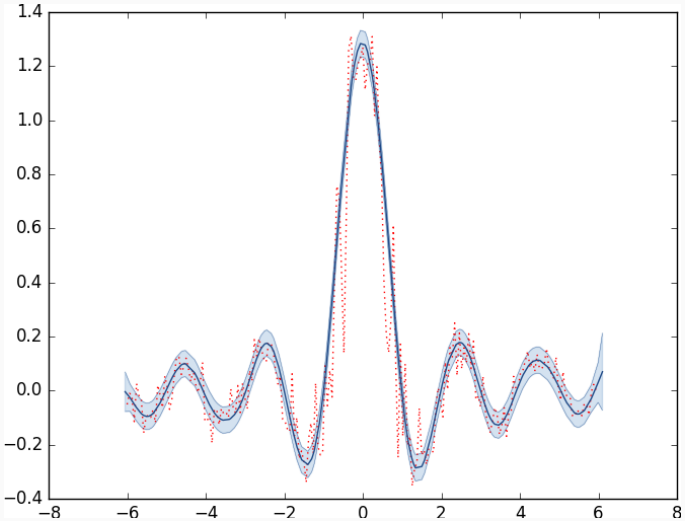
PREDICCIÓN GP COMPLETO DE MÚLTIPLE SALIDA



(a) $\text{sinc}(x) + \sigma$

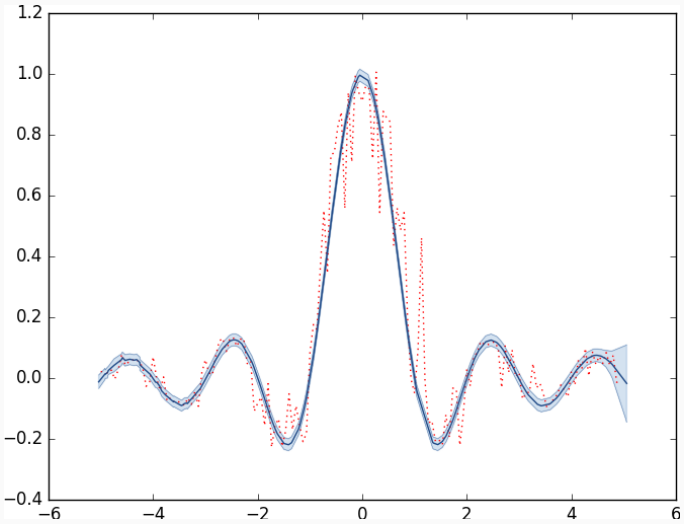


(b) $\text{sinc}(x + \frac{\pi}{4}) + \sigma$

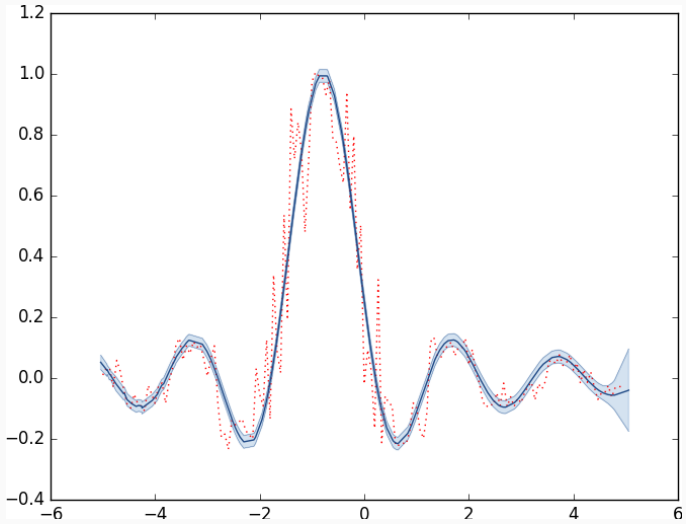


(c) $1{,}3 \times \text{sinc}(x) + \sigma$

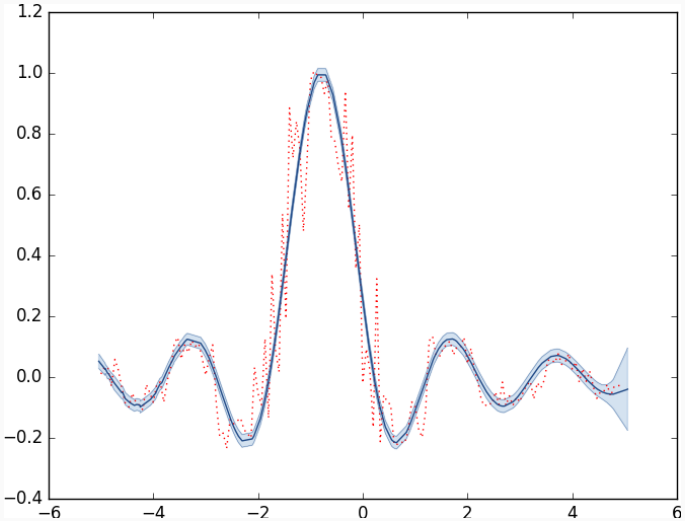
PREDICCIÓN GP FITC DE MÚLTIPLE SALIDA



(a) $\text{sinc}(x) + \sigma$



(b) $\text{sinc}(x + \frac{\pi}{4}) + \sigma$



(c) $1{,}3 \times \text{sinc}(x) + \sigma$

PRUEBAS DE DESEMPEÑO

En estas pruebas se compara el desempeño de la librería usando la regresión completa y la aproximación FITC.

Para cada una de las pruebas, se toma un máximo de 1000 iteraciones para el entrenamiento y se lleva registro de la función de costo, los cálculos previos al gradiente y los cálculos del gradiente en cada una de estas iteraciones.

Después de tomados los tiempos en cada iteración, se promedian y se clasifican.

GP SIN OPTIMIZAR LOS PUNTOS OCULTOS

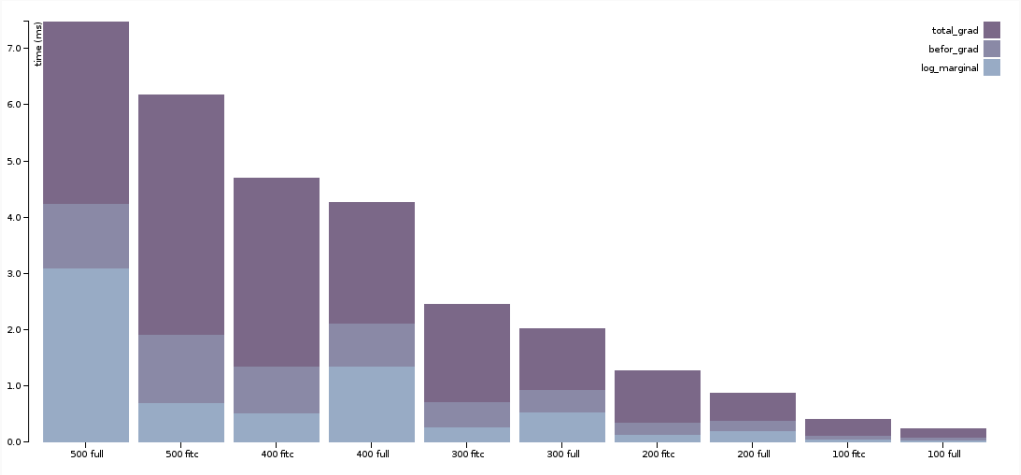


Figura: GP de múltiple salida, de 100 a 500 puntos

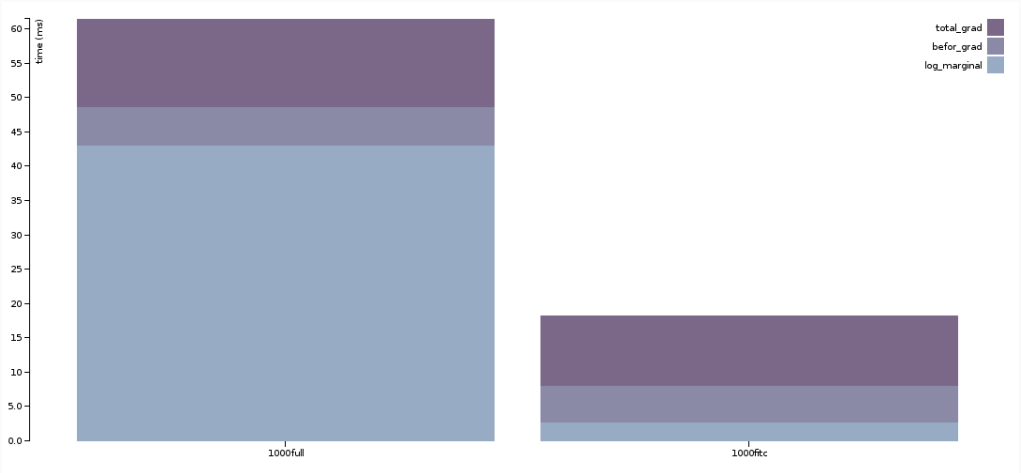


Figura: GP de múltiple salida, de 1000 puntos

GP CON OPTIMIZACIÓN COMPLETA

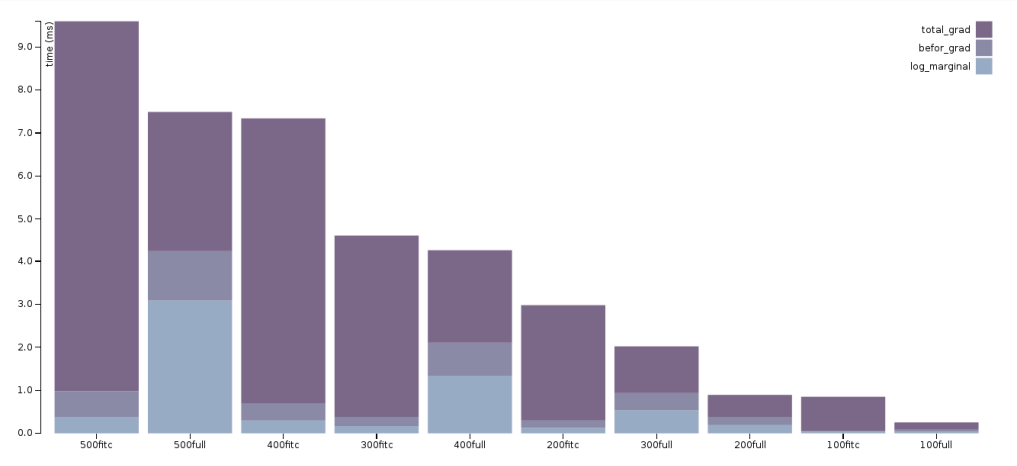


Figura: Prueba de 100 a 500 puntos

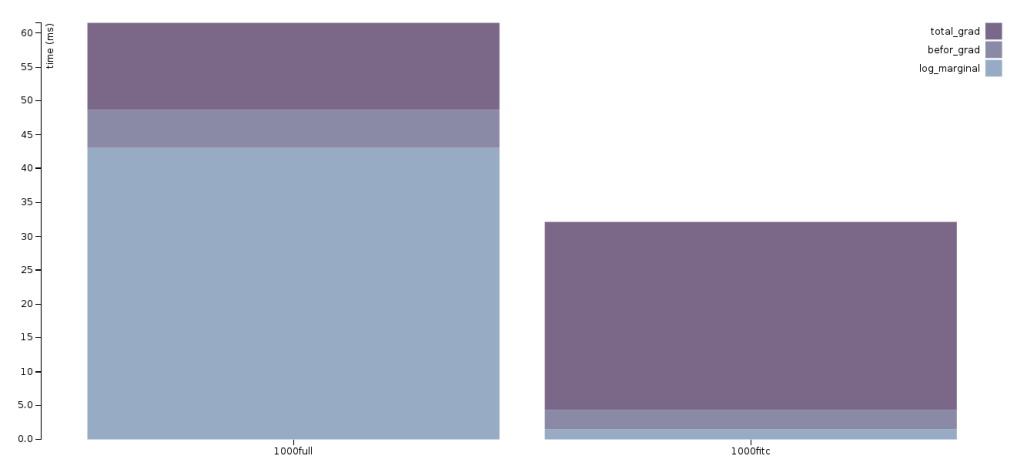


Figura: Prueba de 1000 puntos

CONCLUSIONES

CONCLUSIONES

- Se implementaron todas las rutinas a modo de librería de fácil instalación, de manera que puede ser llamada desde cualquier código en C++ o se pueden crear ”bindings”.
- Se proveen métodos sencillos para crear un kernel de múltiple salida y un objeto de regresión, de manera que el usuario pueda utilizar fácilmente la librería, pero además se proveen métodos más complejos para quienes necesiten mayor flexibilidad a la hora de modelar el proceso gaussiano.

CONCLUSIONES

- Con el propósito de evaluar el desempeño de la librería, se realizó un conjunto de pruebas, evaluando los dos métodos de regresión, FULL GP y FITC. La implementción FITC es asintóticamente más rápida que la implementación FULL GP.

CONCLUSIONES

- El proceso de entrenamiento sin tomar en cuenta los puntos ocultos y el proceso de predicción en la aproximación FITC son significativamente más rápidos que el FULL GP, tal como lo sugería su complejidad teórica.
- Gracias al uso de la librería Armadillo, GPlib corre utilizando múltiples hilos del procesador, lo cual es de gran conveniencia para cumplir las metas de desempeño.
- Debido a la forma en que se implemento la librería, es posible agregar de manera fácil nuevos kernels, tanto los de única salida como los de múltiples.