The background features a complex network of thin grey lines connecting various points, forming a web-like structure. Scattered throughout are several triangles of different sizes and orientations, some with solid grey outlines and others with dashed or dotted lines. The overall aesthetic is technical and modern.

Fundamentals of Bayesian Analysis with PyMC3 and TFP

TW: @charangostation

IG: @sebasarango118

About me



**Lead Data Scientist @
Experimentality**



**MSc. (asp) in Software
Engineering @ EAFIT**



**BSc. in Electronics
Engineering @ UdeA**



Introduction

Statistics & Probability in ML/DL?

- **Every** ML model makes use of **statistics**.
- Libraries and frameworks tend to **mask** its details.
- Strong **prior assumptions** on data.



Do you use...? Then you assume...

Naive Bayes

Variables to be statistically **independent**.

Linear Regression

Error to be **Gaussian distributed** by default in many cases.

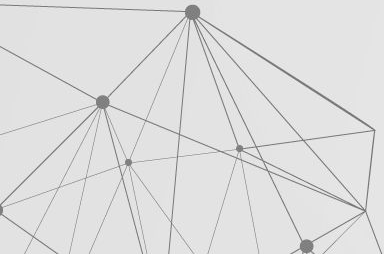
Mean Squared Error

Homoscedasticity
(variance remains constant and finite for assessment)

Pearson Correlation Coefficient
Linearity and **Homoscedasticity**.

Get to learn Statistics!

It might make the difference (*or it might not*)



Bayesian Analysis

Bayesian Analysis

- One **approach** to probabilistic modeling and inference.
- Very close to **human** (natural) decision-making.
- Allows to quantify **uncertainty**.
- Cares about **a priori** assumptions (ML tools use them a lot).
- Prior knowledge is modified by **evidence**.





Bayesian

- More intuitive, less used though.
- Parameters (Θ) are unknown.
- Evidence data (\mathbf{D}) are known.
- Posterior $p(\Theta | \mathbf{D})$ summarizes everything about Θ . **MAP**.
- **Credible** Intervals (uncertainty relates to Θ).



Frequentist

- Less intuitive.
- Parameters (Θ) are not RV.
- Evidence data (\mathbf{D}) are unknown.
- Likelihood $p(\mathbf{D} | \Theta)$ explains the evidence. **MLE**.
- **Confidence** Intervals (uncertainty relates to interval itself).

Probability Rules

$$p(A \cup B) = p(A) + p(B) - p(A \cap B)$$



Prob. of **Union**

$$p(A, B) = p(A \cap B) = p(A | B)p(B)$$



Joint prob.

$$p(A) = \sum_b p(A, B) = \sum_b p(A | B = b)p(B = b)$$



Marginal prob.

$$p(A | B) = \frac{p(A, B)}{p(B)}, p(B) \neq 0$$



Conditional prob.

Bayes' Theorem

The diagram illustrates Bayes' Theorem with the following components and arrows:

- Posterior**: Labeled below $p(\theta | D)$ with a downward arrow.
- Likelihood**: Labeled above $p(D | \theta)$ with an upward arrow.
- Prior**: Labeled above $p(\theta)$ with an upward arrow.
- Normalizing factor**: Labeled below the denominator $\sum_{A_i} p(D | \theta_i)p(\theta_i)$ with a downward arrow.

$$p(\theta | D) = \frac{p(D | \theta)p(\theta)}{p(D)} = \frac{p(D | \theta)p(\theta)}{\sum_{A_i} p(D | \theta_i)p(\theta_i)}$$

Bayes' Theorem

- Bayesian → **Posterior**
- Frequentist → **Likelihood** (misses prior beliefs)
- **Normalizing factor** can be a sum (**discrete**) or an integral (**continuous**)
- It is difficult sometimes to calculate it analytically → **Computational methods**



Computational Methods

Variational Bayes Inference

- Provides exact, analytical solutions
- Locally optimal

Monte Carlo algorithms

- Draw statistically independent samples (no autocorrelation)

Markov Chain Monte Carlo algorithms

- Metropolis-Hastings → Gibbs Sampling | Hamiltonian MC ...



Probabilistic Programming

Probabilistic Programming Tools for Python



PyMC3

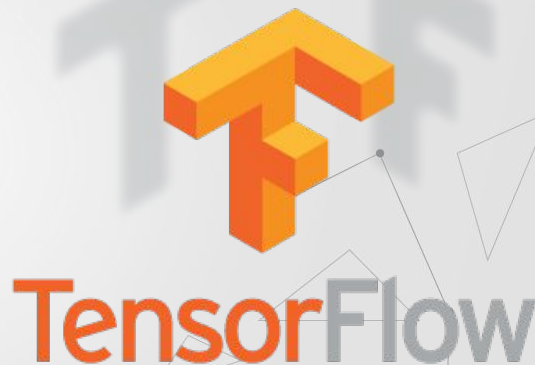


TensorFlow Probability

- **Library** for probabilistic reasoning and statistical analysis in **TensorFlow**.
- Supports major **TF features**: *Eager* mode, HW acceleration, autodiff, vectorization...

Low-level blocks: Distributions | Bijectors

High-level blocks: MCMC | Probabilistic layers |
Structural time series | Edward2



PyMC3

- **Library** for probabilistic reasoning and statistical analysis over **Theano**.
- Supports **MCMC** and **Variational Bayes** inference methods.
- Easy-to-use **Python** user interface.
- Defines models as Python **context** managers.



Example: Bayesian Linear Regression

Conclusions

Conclusions

- **Simplicity:** PyMC3 is designed to provide a more straightforward user experience than TFP.
- **Control:** TFP is more open to tuning and modifications.
- **Support:** TFP has a better community support.
- **Computation:** TFP relies on top features of TF.





Thank you!

TW: @charangostation

IG: @sebasarango118

EM: sebasarango1180@gmail.com