UNIVERSIDAD DE GRANADA

GENERACIÓN DE DATOS SINTÉTICA A PARTIR DE UN SISTEMA IOT DE DETECCIÓN DE VEHÍCULOS UTILIZANDO MODELOS TRANSFORMER CON ATENCIÓN EN PYTHON

RICARDO RUIZ FERNÁNDEZ DE ALBA

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores

María Bermúdez Edo

FACULTAD DE CIENCIAS

E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 17 de junio de 2024

ÍNDICE GENERAL

I. FUNDAMENTOS MATEMÁTICOS ELEMENTALES 4

1. SECCIÓN PRIMERA 5

2. SECCIÓN SEGUNDA 7

II. PARTE DE INFORMÁTICA 8

3. SECCIÓN TERCERA 9

RESUMEN

Nos basamos en el trabajo desarrollado en [VSP+].

Occaecati expedita cumque est. Aut odit vel nobis praesentium dolorem consequatur. Et cumque quia recusandae fugiat earum repellat porro. Earum et tempora vel voluptas. At sed animi qui hic eaque

Parte I

FUNDAMENTOS MATEMÁTICOS ELEMENTALES

En este capítulo se presentarán las definiciones y resultados fundamentales de la teoría de probabilidad que servirán como base para el desarrollo posterior del trabajo.

1

SECCIÓN PRIMERA

Estableceremos la teoría bajo la suposición de que hay un conjunto no vacío Ω, que representa todos los posibles resultados de un experimento aleatorio. Definimos un *evento* como cualquier subconjunto de Ω.

Definición 1 (σ-álgebra)**.** Sea $\mathcal{P}$ partes de Ω. Llamamos σ-álgebra a cualquier $\mathcal{A} \subset \mathcal{P}$ que cumpla:

- $\mathcal{A} \neq \emptyset$.
- Si $A_1, A_2, \ldots, A_n \in \mathcal{A}$ entonces $\bigcup_{i=1}^{n} A_i \in \mathcal{A}$.
- Si $A \in \mathcal{A}$ entonces $A^c \in \mathcal{A}$.

A partir de las propiedades anteriores, deducimos que $\Omega \in \mathcal{A}$ y que $\mathcal{A}$ también es cerrado bajo intersecciones numerables.

Definición 2 (Función de probabilidad)**.** $P : \mathcal{A} \to [0, 1]$ es una función de probabilidad si satisface los tres axiomas:

1. $P(A) \geq 0 \quad \forall A \in \mathcal{A}$.
2. $P(\Omega) = 1$
3. P es σ-aditiva, es decir: dada $\{A_n\}_{n \in \mathbb{N}} \subset \mathcal{A}$ es una sucesión de conjuntos disjuntos dos a dos entonces

$$P\left(\bigcup_{i \in \mathbb{N}} A_i\right) = \sum_{i \in \mathbb{N}} P(A_i).$$

Identificaremos como suceso seguro al evento que siempre ocurre. La primera condición nos asegura que el suceso seguro tiene la probabilidad más alta posible. La segunda condición garantiza que la probabilidad es no negativa. Finalmente, la tercera condición establece que, para un conjunto de sucesos disjuntos, la probabilidad de que ocurra alguno de ellos es igual a la suma de las probabilidades individuales de cada suceso.

Proposición 1. *Toda medida de probabilidad P, cumple:*

- $P(\emptyset) = 0$
- *Dados $A, B \in \mathcal{A}$, entonces $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.*

Demostración. La primera propiedad se deduce de la σ-aditividad de P y de que $\emptyset = \bigcup_{i=1}^{\infty} \emptyset$. La segunda propiedad se deduce de la σ-aditividad de P y de que $A \cup B = A \cup (B \setminus A)$ disjuntos.

$\square$

Definición 3 (Espacio de probabilidad)**.** Definimos como *espacio de medida* a la terna $(\Omega, \mathcal{A}, \mu)$, donde $\mu : \mathcal{A} \to R_0^+$ es una medida en $(\Omega, \mathcal{A})$ y *espacio de probabilidad* a la tupla formada por $(\Omega, \mathcal{A}, P)$, donde P es una medida de probabilidad en $(\Omega, \mathcal{A})$.

2

SECCIÓN SEGUNDA

Sección segunda.

Parte II

PARTE DE INFORMÁTICA

Esta es una descripción de la parte de informática.

3

SECCIÓN TERCERA

El siguiente código es un ejemplo de coloreado de sintaxis e inclusión directa de código fuente en el texto usando `minted`.

```haskell
-- From the GHC.Base library.
class  Functor f  where
    fmap        :: (a -> b) -> f a -> f b

    -- | Replace all locations in the input with the same value.
    -- The default definition is @'fmap' . 'const'@, but this may be
    -- overridden with a more efficient version.
    (<$)        :: a -> f b -> f a
    (<$)        =  fmap . const

-- | A variant of '<*>' with the arguments reversed.
(<**>) :: Applicative f => f a -> f (a -> b) -> f b
(<**>) = liftA2 (\a f -> f a)
-- Don't use \$ here, see the note at the top of the page

-- | Lift a function to actions.
-- This function may be used as a value for `fmap` in a `Functor` instance.
liftA :: Applicative f => (a -> b) -> f a -> f b
liftA f a = pure f <*> a
-- Caution: since this may be used for `fmap`, we can't use the obvious
-- definition of liftA = fmap.

-- | Lift a ternary function to actions.
liftA3 :: Applicative f => (a -> b -> c -> d) -> f a -> f b -> f c -> f d
liftA3 f a b c = liftA2 f a b <*> c


{-# INLINABLE liftA #-}
{-# SPECIALISE liftA :: (a1->r) -> IO a1 -> IO r #-}
{-# SPECIALISE liftA :: (a1->r) -> Maybe a1 -> Maybe r #-}
{-# INLINABLE liftA3 #-}
```

```
{-# SPECIALISE liftA3 :: (a1->a2->a3->r) -> IO a1 -> IO a2 -> IO a3 -> IO r #-}
{-# SPECIALISE liftA3 :: (a1->a2->a3->r) ->
                                Maybe a1 -> Maybe a2 -> Maybe a3 -> Maybe r #-}

-- | The 'join' function is the conventional monad join operator. It
-- is used to remove one level of monadic structure, projecting its
-- bound argument into the outer level.
join              :: (Monad m) => m (m a) -> m a
join x            =  x >>= id
```

Vivamus fringilla egestas nulla ac lobortis. Etiam viverra est risus, in fermentum nibh euismod quis. Vivamus suscipit arcu sed quam dictum suscipit. Maecenas pulvinar massa pulvinar fermentum pellentesque. Morbi eleifend nec velit ut suscipit. Nam vitae vestibulum dui, vel mollis dolor. Integer quis nibh sapien.

BIBLIOGRAFÍA

[VSP+] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Publisher: [object Object] Version Number: 7.