Fundamentos Matemáticos del Machine Learning

Manuel Gijón Agudo

Índice

1. Introducción **2**

2. Probabilidad **3**

2.1. Introducción 3

2.2. Conceptos básicos 3

2.3. Distribuciones discretas 3

2.3.1. Bernulli, $B(1, p)$ 3

2.3.2. Binomial, $B(n, p)$ 3

2.3.3. Binomial Negativa, $BN(r, p)$ 3

2.3.4. Multinomial 3

2.3.5. Chi Cuadrado de Pearson, χ^2_n 4

2.3.6. T de Student, t_n 4

2.3.7. F de Fisher-Snedecor, F_{n_1,n_2} 4

2.4. Teoremas y resultados 4

3. Grafos **5**

3.1. Introducción 5

3.2. Conceptos básicos 5

4. Word2Vect **6**

4.1. Introducción 6

4.2. The Skip-Gram model 6

4.2.1. Parametrización del modelo Skip-Gram 6

4.3. The Continuous Bag-of-Words Models (CBOW) 7

Referencias **8**

1. Introducción

2. Probabilidad

2.1. Introducción

2.2. Conceptos básicos

2.3. Distribuciones discretas

2.3.1. Bernulli, $B(1, p)$

2.3.2. Binomial, $B(n, p)$

2.3.3. Binomial Negativa, $BN(r, p)$

2.3.4. Multinomial

La **distribución multinomial** una generalización de la distribución binomial.

La distribución binomial es la probabilidad de un número de éxitos en N sucesos de Bernoulli independientes, con la misma probabilidad de éxito en cada suceso. En una distribución multinomial, el análogo a la distribución de Bernoulli es la distribución categórica, donde cada suceso concluye en únicamente un resultado de un número finito K de los posibles, con probabilidades $p_1, p_2, ..., p_k$ (tales que $p_i \geq 0$ para $i \in [0, k]$ y $\sum_{i=1}^{k} p_i = 1$); y con n sucesos independientes.

Sea la variable aleatoria X_i, que indica el número de veces que se ha dado el resultado i entre los n sucesos. El vector $X = (X_1, ..., X_k)$ sigue una distribución multinomial con parámetros n y p, donde $p = (p_1, ..., p_k)$.

- Parámetros:
  - $n \in \mathbb{N}$: número de pruebas.
  - $p_1, ..., p_k$: probabilidad de un suceso concreto, tales que $\sum p_i = 0$.
- Dominio: $X_i \in \{0, ..., n\}$ tales que $\sum X_i = n$.
- Función de densidad:
$$\frac{n!}{x_1! ... x_k!} p_1^{x_1} ... p_k^{x_k}$$
- Media: $\mathbb{E}(X_i) = np_i$
- Varianza: $Var(X_i) = np_i(1 - p_i)$
- Covarianza: $Cov(X_i, X_j) = -np_i p_j \ , (i \neq j)$
- Función generadora de momentos:
$$\left(\sum_{i=1}^{k} p_i e^{t_i} \right)^n$$

2.3.5. Chi Cuadrado de Pearson, χ_n^2

2.3.6. T de Student, t_n

2.3.7. F de Fisher-Snedecor, F_{n_1,n_2}

2.4. Teoremas y resultados

3. Grafos

3.1. Introducción

3.2. Conceptos básicos

4. Word2Vect

4.1. Introducción

Word2Vect es un grupo de modelos de software creados por Tomas Mikolov (entre otros, [TM]) usados para la producción de *word embeddings*.

4.2. The Skip-Gram model

Dado un **conjunto de palabras (corpus of words)** ω y su **contexto (context)** $\mathfrak{C}$, consideramos las probabilidades condicionadas $p(\mathfrak{C}|\omega)$, y dado un **cuerpo de Texto (corpus Text)**, el objetivo es definir un conjunto de parámetros θ de $p(\mathfrak{C}|\omega;\theta)$ tal que maximice las probabilidades del corpus ω:

$$\arg\max_{\theta} \prod_{\omega\in\text{Texto}} \left[\prod_{c\in C(\omega)} p(c|\omega;\theta) \right] \tag{1}$$

en esta ecuación, $C(\omega)$ es el conjunto de palabras del contexto ω. Alternativamente:

$$\arg\max_{\theta} \prod_{(\omega,c)\in\mathfrak{D}} p(c|\omega;\theta) \tag{2}$$

donde $\mathfrak{D}$ es el conjunto de todos los pares palabra y contexto extraídos del texto.

4.2.1. Parametrización del modelo Skip-Gram

Una aproximación viene del la literatura rederente a los modelos de redes neuronales relativos al elnguaje y a modelos de la probabilidad condicional que utilizan *soft-max*[1]:

$$p(c|\omega;\theta) = \frac{e^{v_c\cdot v_\omega}}{\sum_{c'\in\mathfrak{C}} e^{v'_c\cdot v_\omega}} \tag{3}$$

donde v_c y v_ω pertenecen a $\mathbb{R}^d$ son representaciones vectoriales para c y ω respectivamente y $\mathcal{C}$ es el conjunto de todos los contextos disponibles. Los parámetros θ son v_{c_i}, v_{ω_i} para $\omega\in\mathcal{V}, c\in\mathfrak{C}, i\in\{1,...,d\}$ (un total del $|\mathfrak{C}|\times|\mathcal{V}|\times d$ parámetros). Nos gustaría fijar los parámetros que maximicen (2).

Ahora aplicamos logaritmos para transformar el producto en una suma:

$$\arg\max_{\theta} \sum_{(\omega,c)\in\mathfrak{D}} \log p(c|\omega) = \sum_{(\omega,c)\in\mathfrak{D}} \left(\log e^{v_c\cdot v_\omega} - \log \sum_{c'} e^{v_{c'}\cdot v_\omega} \right) \tag{4}$$

Bajo todo esto se encuentra la siguiente hipótesis que no explicaremos ahora:

[1] La **función softmax** o **función exponencial normalizada** es una generalización de la función logística $P(t)=\frac{1}{1+e^{-t}}$. Se emplea para "comprimir" un vector k-dimensional ($\mathfrak{z}$) de valores reales arbitrarios en otro vector ($\sigma(\mathfrak{z})$) también k-dimensional pero con valores en el rango $[0,1]$.

Hipótesis: maximizar (4) nos dará buenos embeddings v_ω $\forall \omega \in V$, en el sentido de que palabras similares producirán vectores similares.

Calcular (4) puede ser computacionalmente muy costoso debido a que calcular el término $p(c|\omega;\theta)$ implica calcular el sumatorio $\sum_{c' \in \mathfrak{C}} e^{v_{c'} \cdot v_\omega}$. Una manera de hacer este cálculo más factible es remplazar el softmax por un *hierarchical softmax*.

4.3. The Continuous Bag-of-Words Models (CBOW)

Referencias

[YGOL] Yoav Goldberg and Omer Levy "word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method" **31** (February 14, 2014)

[XR] Xin Rong "word2vec Parameter Learning Explained" (June 5, 2016)

[TM] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient es- timation of word representations in vector space" **CoRR, abs/1301.3781** (2013)

[SR] Sebastian Raschka "Python Machine Learning" **Packt Publishing Open Source** (2015)