LANGuage IDentification

João Augusto Costa Branco Marado Torres

December 14, 2024

License

Copyright © 2024 João Augusto Costa Branco Marado Torres. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents

List of Figures

Table 1: Rainbow model vocabulary

ID	Word
0	red
1	orange
2	yellow
3	green
4	cyan
5	blue
6	violet

1 Phrases to Numeric Representation

I was really confused about this, but it really is not that complicated.

Before starting to write any code, I wanted to know how can I transform a phrase like the one you are reading right now into `1`s and `0`s because the computer isn't smart enough to know English. So I went to my friend and asked it some questions. It thought me various ways of achieving what I wanted, and between those options I picked the easiest to understand/implement.

1.1 Bag of Words

The idea it's to build a **vocabulary** which is a list of unique words.

Everyone (including us) has its own vocabulary composed by words of the languages we speak.

Our AI model needs a vocabulary too so he can understand some words. Those words might not exist in our vocabulary.

Let's say our model's vocabulary 1 are the words in English for the `7` colors in the rainbow.

Each word in the vocabulary will have a numerical identifier.

It makes sense for us for that identifiers to start at `0` and increment by `1` for each word.

Now our model can represent our phrases into something it understands.

> How?

The question you just asked is represented as a zero column matrix with 7 rows.

The model will try to find words he knows from a phrase and represent it as a column matrix with rows the same as the amount of words in the vocabulary.

Copyright © 2024 João Augusto Costa Branco Marado Torres.

Each entry represents a word in the vocabulary, here is why it's a good idea to identify the words as I said. The value of each entry can be a simple boolean that represent if a word is present in the phrase or not, or the amount of times the word appears in the phrase. The value really represents how important is the that word in the phrase.

The phrase "How?" does not contain any words know by the model.

The phrase "The colors in the *Guiné-Bissau* flag are red, yellow, green and black in the star." would be represented by the matrix $[\,1\;0\;1\;1\;0\;0\;0\,]^T$ and the phrase "What came first, the orange fruit or the orange color?" could be represented by $[\,0\;2\;0\;0\;0\;0\;0\,]^T$ if we count the amount of occurrences of the word in the phrase.

The way I implemented the BoW was by reading a file and collect every single word from it separated by white spaces, dashes, punctuation, parenthesis or quotation marks and then spit to `stdout` a formatted vocabulary, so you can pipe it later or write it to a file. Words are case-sensitive, and it's possible to create the vocabulary in alphabetic order.

My implementation can be found in `/src/commands/vocab/bow.zig`.

During the conversation, "tokenization" was mentioned, and it might be cool for you to take a look at it.

1.2 Other ways

- TF-IDF;
- One-Hot Encoding of Words;
- Embedding-like Approach;
- Sub-word tokenization.

2 Neuron

As explained at the start of chapter 3.1 of [Bis06], the base for the simplest supervised linear regression models will look like (1)

$$\begin{aligned} y(\mathbf{x}, \mathbf{w}) &= (1)b + w_0 x_0 + w_1 x_1 + \ldots + w_D x_D = \\ &= (1)b + \sum_{j=0}^{D} (w_j x_j) \end{aligned} \tag{1}$$

where $\mathbf{x} = [\,x_0 \;\cdots\; x_D\,]^T$, a linear function of the parameters $\mathbf{w} = [\,w_0 \;\cdots\; w_D\,]$ and bias b. We can also make the bias part of the $\mathbf{w}$ and at the same position in $\mathbf{x}$ put the

Copyright © 2024 João Augusto Costa Branco Marado Torres.

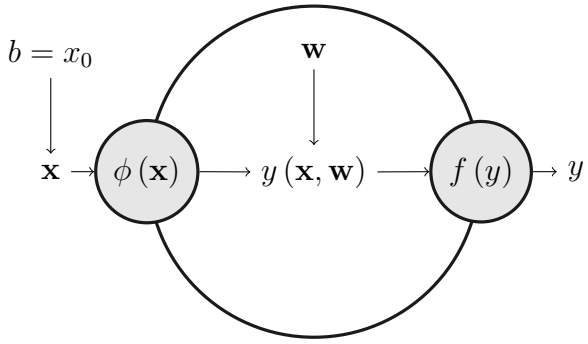

Figure 1: A visual representation of a neuron.

value 1 and have the equation like (2)

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{x} \cdot \mathbf{w} \tag{2}$$

a simple matrix multiplication. Both matrixes have the same amount of elements D.

There is also the notion of applying what is called as **basis function** to the $\mathbf{x}$ parameters before everything. So there will be a matrix of basis functions $\phi = \begin{bmatrix} \phi_0 & \cdots & \phi_D \end{bmatrix}^T$ where $b = x_i$ and $\phi_i(b) = 1$.

If that's the case the function will look like this (3).

$$\begin{aligned} y(\mathbf{x}, \mathbf{w}) &= \sum_{j=0}^{D} (w_j \phi_j(x_j)) = \\ &= \phi(\mathbf{x}) \cdot \mathbf{w} \end{aligned} \tag{3}$$

The functions can be seen as a way to pre-process and extract the important parts of the input This list can have nonlinear functions making $y(\mathbf{x}, \mathbf{w})$ nonlinear too.

In case we are working with a classification model, we might want to introduce an **activation function** $f(\cdot)$ that after all the computations in $y(\mathbf{x}, \mathbf{w})$ (3) transforms that result into a value corresponding to being the probability of $\mathbf{x}$ being of a certain class.

And this will be what I define as an artificial neuron (4). You can see a visual representation of it on page 4.

$$y(\mathbf{x}, \mathbf{w}) = f(\phi(\mathbf{x}) \cdot \mathbf{w}) \tag{4}$$

Because I couldn't find a simple way of doing anonymous functions in my language of choice, my implementation does not allow specifying basis functions

Copyright © 2024 João Augusto Costa Branco Marado Torres.

nor activation functions so every basis function can be thought as being the identify function $\phi(x) = x$ and the activation function can be toggled between the identify and sigmoid (5).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

3 Feed-Forward Neural Network

The idea now, as explained in chapter 5.1 of [Bis06], is to train neurons so that their weight values are good.

Because a neuron is really just a function, a neuron can be a basis function of another neuron, starting a network.

A network is composed of layers, each with its own number of neurons.

Each of neuron will create a value called **activation** represented by a_j (6) where j is the neuron number (we can think of a layer as a column matrix of neurons), $\mathbf{X}$ is the function input or the activations from the layer before after they passed through an activation function and D is the amount of row in $\mathbf{x}$ -1.

Let's define a way to represent a matrix row:

$$A = \begin{bmatrix} a_0 \\ \vdots \\ a_N \end{bmatrix} \in \mathbb{R}^{N \times M}, a_i \in \mathbb{R}^M, i \in \{x \in \mathbb{N} : x \leq N\}$$

$$A_{i*} = a_i$$

$$\begin{aligned} a_j &= f\left(\sum_{i=1}^{D} (w_{ji} x_i) + (1) b_j\right) \\ &= f(\mathbf{x} \cdot W_{j*}) \end{aligned} \tag{6}$$

The last layer will be the one giving the outputs $\mathbf{y}$.

If a neural network has n layers, the mathematical representation for the output k will be something like (9). $W^{(n)}$ are the weights of that layer. For this to work, because it's just matrix multiplications, we have to be aware of the dimensions of the matrixes. This is called **forward propagation**.

$$Y_0 = \mathbf{x} \tag{7}$$

$$Y_n = f_n\left(Y_{n-1}^T \cdot w^{(n)}\right), n > 0 \tag{8}$$

Copyright © 2024 João Augusto Costa Branco Marado Torres.

$$
\begin{aligned}
y_k(\mathbf{x}, \mathbf{w}) &= f_n \left(\sum_{i_n=0} w_{ki_n}^{(n)} \cdots f_2 \left(\sum_{i_2=0} w_{i_3 i_2}^{(2)} f_1 \left(\sum_{i_1=0} w_{i_2 i_1}^{(1)} x_{i_1} \right) \right) \cdots \right) = \\
&= f_n \left(\cdots f_2 \left(f_1 \left(\mathbf{x} \cdot w^{(1)} \right)^T \cdot w^{(2)} \right)^T \cdots \cdot w^{(n)} \right) = \\
&= Y_n
\end{aligned}
\tag{9}
$$

Layers and neural network representation can be found `/src/utils/neural.zig`.

References

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 9780387310732.

Copyright © 2024 João Augusto Costa Branco Marado Torres.