Journal of Electronic Research and Application

Research Article

Data Processing and Artificial Neural Network under Big Data

Junjie Shen

Clarkson Secondary School, Canada

Publication date: March, 2020

Publication online: 31 March, 2020

****Corresponding author:*** Junjie Shen, wangchunlian@xyzrgroup.com

1 Introduction

1.1 Decoding of words

The origin of characters is that human beings can't remember all things, so they must be recorded to convey and inherit. Character is essentially a kind of coding for things. As a kind of text, numbers are concise, easy to understand, and convenient to write and verify. Understanding sentences is a decoding process. With the development of computer science, the question arises: can a computer understand language and apply it to life? For example, the translation is actually the decoding and recoding of words.

In previous studies, researchers focused on exploring whether computers can think and calculate like human brains. This needs to follow the grammar rules, word by word correspondence analysis. But this method is effective for short sentences and simple sentences, and it is very difficult to apply it to long and difficult sentences. At the same time, with the development of language, the different meanings and contexts of words also have a great impact on the decoding.

With the development of civilization, the decoding analysis of regular sentences is gradually replaced by statistics, because the semantic analysis of sentences is largely affected by the context. Therefore, the mathematical model is applied to decoding analysis.

1.2 Statistical language model

This model is applied to speech recognition. It is to judge whether a sentence conforms to grammar rules and expresses correctly. Its most fundamental principle is to calculate the probability of a certain collocation for some kind of words in daily life, and then select the group with the highest probability from a variety of collocations.

S stands for a sentence. w_1, w_2, $w_3...w_n$ is the word that makes up the sentence (n is the sentence length). We will use such a model to estimate P(S) (the probability of the occurrence of a certain collocation of words in daily life).

$P(S)=P(w_1,w_2,w_3...w_n)$

The conditional probability formula in probability calculation can be rewritten as follows:

$P(w_1,w_2,w_3...w_n)=P(w_1)*P(w_2|w_1)*P(w_3|w_1,w_2)...*P(w_n|w_1,w_2...w_{n-1})$

Where P(w1) represents the occurrence probability of the first word w_1 in daily life, $P(w_2/w_1)$ represents the occurrence probability of the second word w_2 in daily life after the first word w_1 is known. However, for such calculation, when the number of words is smaller, it is feasible; when the number of words increases, the difficulty of calculation will multiply, which will exceed the level of existing computer calculation. At this time, Markov hypothesis is applied. Markov hypothesis: suppose that the probability of the occurrence of any word w_i is only related to the word w_{i-1} in front of it, $P(S)=P(w_1)*P(w_2|w_1)*P(w_3|w_2)...*P(w_i|w_{i-1})$. In this way, the amount of computation is greatly reduced, which we call binary model. The next problem is how to estimate the conditional probability. According to the formula of conditional probability:

For $P(w_i|w_{i-1})=\frac{P(w_{i-1},w_i)}{P(w_{i-1})}$, the methods to estimate the joint probability $P(w_{i-1},w_i)$ and the edge probability

$P(w_{i-1})$ are as follows. Because there are a large number of corpora, we only need to count that how many times of $\#(w_{i-1}, w_i)$ w_{i-1}, w_i appears adjacent to each other in the statistical text, and how many times of $\#(w_{i-1})$ w_{i-1} itself appears in the same text, and then we divide the two numbers by the size# of the corpus to get the relative frequency of these words or two-tuple:

$$P(w_{i-1},w_i) \approx \frac{\#(w_{i-1},w_i)}{\#} \quad P(w_{i-1}) \approx \frac{\#(w_{i-1})}{\#}$$

Then: $P(w_i|w_{i-1}) \approx \frac{\#(w_{i-1},w_i)}{\#(w_{i-1})}$

With the help of corpus and this formula, we can effectively calculate the occurrence frequency of a text sequence in life.

1.3 High-order language model

In the previous discussion, we assumed that the word w_i in the text is only related to the first word. We call this model a binary model. If we need to get more accurate results, we need to assume that the word w_i in the text is only related to the previous N-1 words, but not to the more previous words, that is:

$P(w_i|w_1, w_2,... w_{i-1}) = P(w_i|w_{i-N+1}, w_{i-N+2}, ... w_{i-1})$

This hypothesis is called N-1 Markov hypothesis. The corresponding language model is N-gram model. The unigram model of N=1, that is, the probability of text occurrence is context independent. The binary model of N=2, that is, the probability of the occurrence of the current word is related to the previous word. Due to the large amount of calculation, N is generally taken as 3 in life, and higher-order models are rarely used.

1.4 Model training, zero-probability problem and smoothing method

When using the model, we call all the conditional probabilities in the model that need to be calculated as parameters. Through the analysis and calculation of corpus statistics, the process of obtaining these parameters is called model training. In the previous discussion, the model training method seems very simple. But the question is, what if the number of consecutive occurrences is zero for a pair of words w_{i-1}, w_i? Does it mean that $P(w_i|w_{i-1}) = 0$? Or is it possible to get $P(w_i|w_{i-1}) = 1$, if both of $\#w_{i-1}$, $\#w_{i-1}$ are 1?

The way to solve this problem is to increase the number of statistical samples. But even for very large statistical samples, P = 0 will still occur. So we introduce the Goode Turing method to solve this problem: for the event that does not occur, we cannot think that the probability of its occurrence is zero, thus we need to allocate a small proportion to them from the total amount. This way the sum of the probabilities of the visible events is less than one. Therefore, we should reduce the occurrence probability of all the collocations in the corpus, which is based on the principle of "the more untrustworthy, the more statistical discount". Goode Turing estimation formula: suppose that in the corpus, there are N_r words with r times with the occurrence and N_0 words without occurrence. The size of corpus is N:

$$N = \sum_{r=1}^{\infty} N_r$$

The relative frequency of r-occurrence words in corpus is r/R. If we do not do any optimization, we use this relative frequency as the probability estimation of these words.

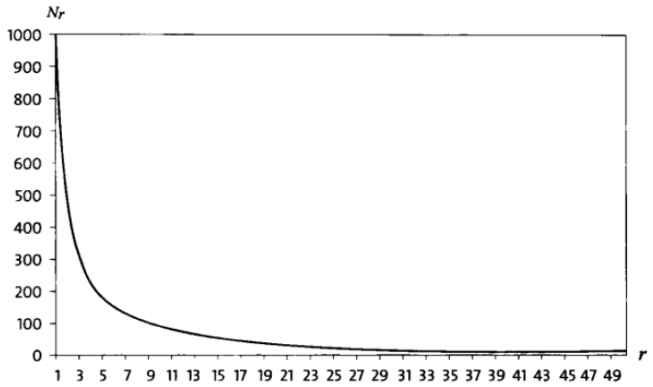
When r is smaller, the statistics may not be reliable. Therefore, d_r (a smaller number of times than r) is used instead of r. According to Goode Turing estimates and following the below formula, d_r is calculated as follows:

$d_r=(r+1) \times N_{r+1}/N_r$

Obviously:

$$\sum_r d_r \cdot N_r = N$$

Generally speaking, the number of words appearing once is more than that appearing twice, and the number of words appearing twice is more than that appearing three times. This law is called Zipf's law. The following figure shows the relationship between the number of words N_r that appear r times and r in a small language database.



It can be seen that the larger r is, the smaller the number of words N_r is, that is, $N_{r+1} < N_r$. Therefore, in general, $d_r<r$, but $d_0 >0$. This gives a very small non-zero value to the words that do not appear, thus solving the problem of zero probability. In practical operation,

the frequency of words whose occurrence times exceed a certain threshold is not adjusted downwards, only the words whose occurrence times are lower than this threshold are adjusted downwards.

2 Method of word segmentation

In order to further process natural language, first of all, we need to segment sentences by the method of word segmentation.

Using the word segmentation method of statistical language model, through several mathematical formulas it can be simply summarized as follows: suppose a sentence S has several word segmentation methods:

$$A_1, A_2, A_3, \cdots, A_k$$

$$B_1, B_2, B_3, \cdots, B_m$$

$$C_1, C_2, C_3, \cdots, C_n$$

Among them, A_1, A_2,...B_1...C_1... are all Chinese words. With different word segmentation methods, we can see that there will be different numbers of words. The most common word in our daily life is the best word segmentation method. If $A_1,A_2,A_3...A_k$ is the best word segmentation method, it should be:

$$P(A_1, A_2, A_3, \cdots, A_k) > P(B_1, B_2, \cdots, B_m)$$

$$P(A_1, A_2, A_3, \cdots, A_k) > P(C_1, C_2, \cdots, C_n)$$

Therefore, we only need to use the statistical language model mentioned before to calculate the probability of sentence occurrence of each word segmentation method. Then we can find out the most probable one, and acquire the best word segmentation method.

3 Covert Markov Model

3.1 Communication model

The natural language is the communication tool, and the connection between language and communication is natural. As mentioned at the beginning, the essence of communication is a process of encoding, decoding and transmission.

Now we abstract the model of language translation: when $o_1,o_2,o_3...$ are known and the conditional probability $P(s_1,s_2,s_3...|o_1,o_2,o_3...)$ is the largest, we can get $s_1,s_2,s_3....$

$$s_1, s_2, s_3, \cdots = \underset{all\ s_1,s_2,s_3,\cdots}{\text{Arg Max}}\, P(s_1, s_2, s_3, \cdots | o_1, o_2, o_3, \cdots)$$

By using Bayes formula, this formula can be equivalently transformed into:

$$\frac{P(o_1, o_2, o_3, \cdots | s_1, s_2, s_3, \cdots) \cdot P(s_1, s_2, s_3, \cdots)}{P(o_1, o_2, o_3, \cdots)}$$

In the process of Chinese to English translation, $P(o_1,o_2,o_3...|s_1,s_2,s_3...)$ indicates the probability that the English text is just translated into the translated Chinese text when the English language has been known; $P(s_1,s_2,s_3..)$ indicates the possibility that the English text $s_1,s_2,s_3..$ itself is a reasonable sentence; $P(o_1,o_2,o_3...)$ indicates the rationality of the translated Chinese text.

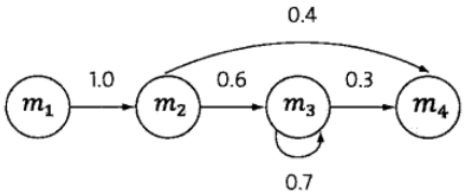
Once the information $o_1,o_2,o_3...$ is generated, it will not change. At this time, $P(o_1,o_2,o_3..)$ is a negligible constant, so the above formula can be equivalent to:

$$P(o_1, o_2, o_3, \cdots | s_1, s_2, s_3, \cdots) \cdot P(s_1, s_2, s_3, \cdots)$$

The below will discuss how to estimate the formula with the covert Markov model

3.2 Covert Markov model

The covert Markov model is developed from Markov chain. So let's discuss Markov chains first.



The above figure is a Markov chain, with four circles representing four states. Each edge represents a possible state transition, and the weight on the edge is the transition probability. For example, there is only one edge from m_1 to m_2 and the weight is 1, and this explains m_1 can only be converted to m_2; there are two edges from the beginning of m_2, and one edge points to m_3 and the weight is 0.6; it means that the conversion probability from m_2 to m_3 at the next moment is 60%; similarly, the probability of conversion to m_4 is 40%. That is, $P(s_t+1 = m3|s_t = m2) = 0.6$, $P(s_t+1 = m4|s_t = m2) = 0.4$.

The covert Markov model is an extension of Markov chain: the state s_t at any time t is invisible. So the observer cannot predict the transition probability by observing a state sequence s_1, s_2, s_3.... But the covert Markov model will output a symbol o_t at every time t, and o_t is only related to s_t. This is called the independent output hypothesis.

Based on Markov hypothesis and independent output hypothesis, we can calculate the probability that a specific state sequence s_1, s_2,s_3... produces

output symbols $o_1,o_2,o_3..$, that is, each English word is translated into corresponding Chinese words as required: (1)

$$P(s_1, s_2, s_3, \cdots, o_1, o_2, o_3, \cdots) = \prod_t P(s_t|s_{t-1}) \cdot P(o_t|s_t)$$

Now take

$$P(o_1, o_2, o_3, \cdots | s_1, s_2, s_3, \cdots) = \prod_t P(o_t|s_t)$$

$$P(s_1, s_2, s_3, \cdots) = \prod_t P(s_t|s_{t-1})$$

into the result obtained from the previous Bayesian formula, and get (1). This shows that the decoding problem can be solved by the covert Markov model.

4 Web crawler

For the Internet, the methods mentioned above can be used to record web pages. In the Internet, every web page can be regarded as a node, and the hyperlink in the web page can be regarded as the path connecting each node. Using the graph traversal algorithm, all web pages are accessed and saved. The program of this function is called "Web Crawler". Starting from the homepage of a website, the web crawler first downloads the webpage, and then analyzes the hyperlinks contained in the webpage, and next finds the way. Again, it continues to visit, download, analyze, and then you can visit and save all the websites on the whole Internet. Of course, many pages may lead to the same page. To avoid duplicate downloads, you need to use a list called a "Hash table" to record which websites have been visited.

5 Determine the relevance between web pages and query results

When we know how to download web pages, we need to determine the relevance between web pages and query results after indexing. For example, when we query the "Application of atomic energy", any search engine can provide hundreds of thousands or even millions of relevant pages, but we need the most relevant pages with keywords. At this point, we need a model to measure the relevance between web pages and query results.

I mentioned Chinese word segmentation before. "Application of atomic energy" can be divided into "atomic energy", "of", and "application" The first thing we are quite sure is that the more keywords appear, the more relevant the web page is. However, this is not fair for a web page with fewer words. We should consider frequency instead of occurrence totality. The quotient obtained by dividing the number of keywords by the total number of words in the web page is called "keyword frequency" or "term frequency" (TF for short). Add the TF of each keyword to get the keyword frequency of "Application of atomic energy" in this web page.

Generally speaking, if a query contains N keywords $w_1, w_2...w_N$. The occurrence frequency of these words in a website is $TF_1, TF_2, TF_3..TF_N$. Then the relevance of such pages can be recorded as:

$R=TF_{1+}TF_{2+}TF_{3+}TF_N$.

But we will find that "of" may account for a large part of the above keywords. But it has no substantial contribution to the screening of web pages. We call such words "stop words". At the same time, we also realize that different keywords have different proportional contributions to the selection of web pages. At this time, we need to give each word a certain weight. Weighting authorization needs to be based on the following two rules:

(1) The stronger the ability of a word to predict a topic, the greater the weight; otherwise, the smaller the weight.

(2) "Stop words" have a weight of 0.

Suppose a word w has appeared in D_w pages, the larger the D_w is, the smaller the weight of w is. We can use the inverse document frequency index (IDF) to measure the weight of each word.

$IDF = \log_2(\frac{D}{D_w})$ where D is the number of all websites and D_w is the number of websites where the word w appears. Suppose there are 1 billion Chinese websites, and the professional word "atomic energy" has appeared in 2 million websites, then its weight IDF = $\log_2(500) = 8.96$. The stop word "of" has appeared in all web pages, so its weight IDF = $\log_2(1)= 0$.

After the introduction of IDF, the formula for calculating the correlation can be transformed as follows:

$R = TF_1 * IDF_1+TF_2 * IDF_2 +... TF_N * IDF_N$

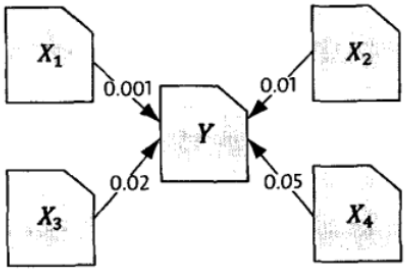
6 Web page ranking technology of PageRank - Google's democratic voting

For most people's queries, search engines can provide thousands of results. How to put the results that users want to see in first-batch place decides the quality of a search engine. In general, the ranking of search

results depends on the quality of the web page and the relevance between the web page and search keywords. In the previous chapters, we have shown how to determine the relevance between a web page and search keywords. Next, we'll show you how to determine the quality of a web page.

When calculating the quality of a web page, we often use the form of "democratic voting": in the Internet, if a web page is linked by many other web pages, it is generally recognized and trusted, then its ranking should be high and its quality should be good, finally the above is the core idea of PageRank.

However, just like voting in a company, people with different shares should have different weights. So we should do the same in the process of calculating the quality of web pages, so that links from different web pages have different weights. For example: we know that the quality of a web page Y should be related to all the web pages pointing to it, and we set a weight for each web page pointing to it, as shown in the following figure, then the PageRank of Y=0.001+0.01+0.02+0.05=0.081.



In practice, we need to know the weights of X_1, X_2, X_3, X_4 to calculate the quality of web page Y. However, the weight of X_1, X_2, X_3, X_4 should depend on their own ranking. This leads to a dead cycle, similar to the question of "for chicken or egg, which first appears".

7 Neural network and back propagation algorithm

Artificial Neural Networks (ANNs) is also referred to as Neural Networks (NNs) or Connection Model. It is an algorithmic mathematical model that imitates the behavior characteristics of biological brain neural network, carries out distributed mode and information processing.

7.1 Neuron

Neurons can be regarded as a process of input and output. Neurons and perceptions are essentially the same, except that the activation function of a perceptron is a step function, while the activation function of a neuron is a sigmoid function or a tanh function.

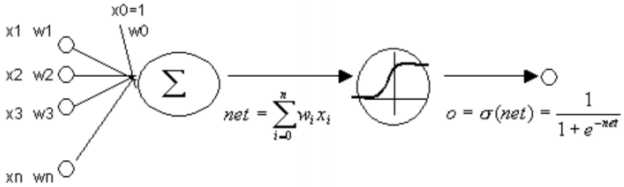
Suppose that the input of neuron is the vector $\vec{x}$, the weight vector is $\vec{w}$, the activation function is sigmoid function, and the output vector is $\vec{y}$, then the following calculation process should be carried out for each neuron vector encountered:

The Sigmoid function is defined as follows:

The figure below shows the calculation of a neuron:

$$y = sigmoid(\vec{w}^T \cdot \vec{x})$$

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$



7.2 Neural network structure

Neural network is a number of neurons connected according to certain rules. The most basic one is the full connected neural network. The below are its rules:

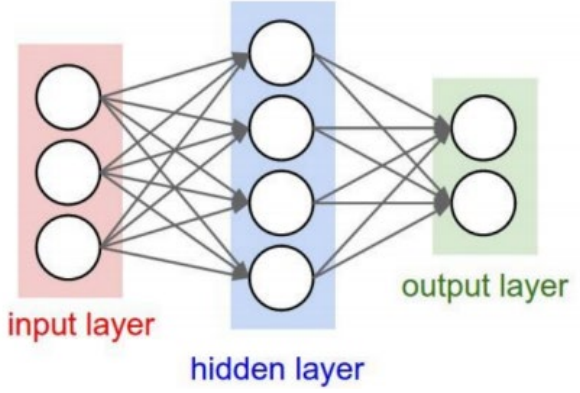
(1)Neurons are arranged from left to right by layers, including input layer, hidden layer and output layer. The input layer is responsible for receiving the input data, and the output layer outputs the calculated data. The hidden layer in the middle is invisible to the outside.

(2)There is no connection between neurons in the same layer

(3) Each neuron in layer N shall be connected with all neurons in layer N-1 (meaning of full connected). The output of neurons in layer N-1 is the input of neurons in layer N

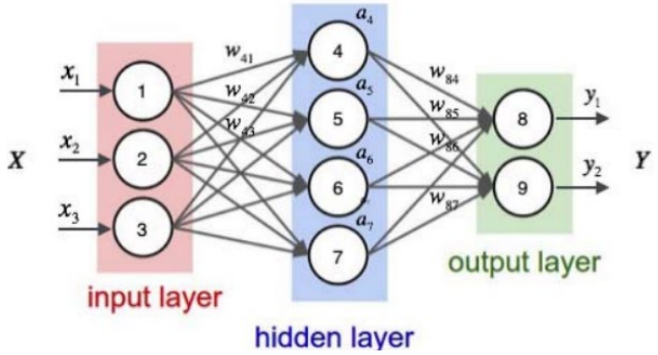
(3)The connection between each layer of neurons and the next layer of neurons has a certain weight.

The figure below shows a fully connected neural network.



7.3 Calculating the output of neural network

Neural network is actually a function from the input vector $\vec{x}$ to the output vector $\vec{y}$ First, we assign the value of each element xi of input vector $\vec{x}$ to each neuron in the input layer of neural network, and then calculate the value of each neuron layer according to the calculation principle until the neuron in the last layer outputs the final result. Here is an example to illustrate the process:



For the three nodes of the input layer, we number them 1, 2, 3; for the four nodes of the hidden layer, we number 4, 5, 6, 7; for the last two nodes of the output layer, we number 8, 9. Because it is a fully connected neural network, every node must have links with all nodes in the upper layer.

Among them, w_{4b} is the bias term of node 4, while w_{41}, w_{42}, w_{43} are the weights of the connection from nodes 1, 2, 3 to 4 respectively. (When numbering weights, we put the number j of the target node in front and the number i of the source node in the back).

Similarly, we can calculate the output values a_5, a_6, a_7 of nodes 5, 6, 7. In this way, you can get the output value y_1 of output layer node 8 by finishing the output values of four nodes in the hidden layer:

$$y_1 = sigmoid(\vec{w}^T \cdot \vec{a})$$
$$= sigmoid(w_{84}a_4 + w_{85}a_6 + w_{86}a_6 + w_{87}a_7 + w_{8b})$$

Then we calculate the output value y_2 of node 9. In this way, the output values of all output layer nodes are calculated completely.

References

[1] Jun Wu. The beauty of mathematics[M]. Posts & Telecommunications Press, 2014