Wenpeng Yin's Blog

word2vec implementations of gradient and expTable

18 Wednesday Dec 2013

Posted by yinwenpeng in deep learning in nlp

≈ 2 Comments

In this blog, I focus on explaining the calculations of "f" and "g" in word2vec C code:

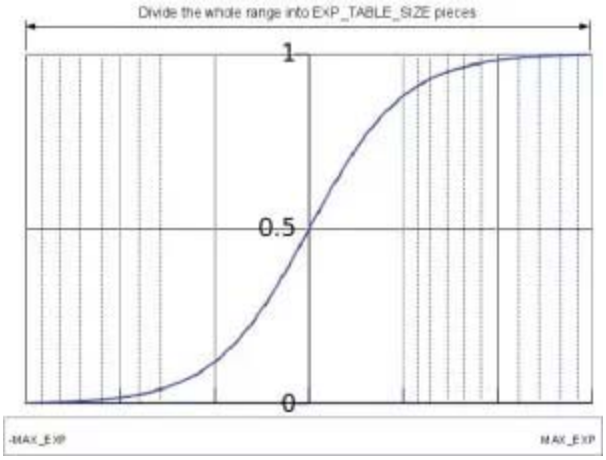
```
else
 {
 // the resulting f is a kind of position
 f = (f + MAX_EXP) * (EXP_TABLE_SIZE / MAX_EXP / 2);
 f = expTable[(int) f];
 }
 // 'g' is the gradient multiplied by the learning rate
 double g = (1 - word.codeArr[i] - f) * alpha;
```

For "f":

It uses a speed-up method to compute the sigmod function:

```
for (i = 0; i < EXP_TABLE_SIZE; i++) {
 expTable[i] = exp((i / (real)EXP_TABLE_SIZE * 2 - 1) * MAX_EXP);
 expTable[i] = expTable[i] / (expTable[i] + 1);
 }
```

Actually, this solution is very simple to understand.



Easy to find that sigmoid(x) values almost keep unchanged when x is too large or too small. Let's assume sigma(x)=1 when x>MAX_EXP, and sigma(x)=0 when x<-MAX_EXP. The code splits the range [-MAX_EXP, MAX_EXP] into EXP_TABLE_SIZE pieces. In each piece, we assume the sigma(x) values are the same.

So, in the initialization of expTable, when i=0, that equals to we choose the first piece, where original x=-MAX_EXP; similarly, when i=EXP_TABLE_SIZE-1, the last piece is selected.

Based on this conversion, each time when we get the raw "f" value, we only need to figure out which piece this value should be located. That is implemented by

```
1  f = (f + MAX_EXP) * (EXP_TABLE_SIZE / MAX_EXP / 2);
```

Namely, each piece enjoys $\frac{2*MAX\_EXP}{EXP\_TABLE\_SIZE}$, and the distance of raw "f" to the left border "-MAX_EXP" is $f-(-MAX\_EXP)=f+MAX\_EXP$. Hence, the index of target piece should be $\frac{f+MAX\_EXP}{\frac{2*MAX\_EXP}{EXP\_TABLE\_SIZE}}=(f+MAX\_EXP)*(EXP\_TABLE\_SIZE/MAX\_EXP/2)$.

For g:

As my another blog: https://yinwenpeng.wordpress.com/2013/09/26/hierarchical-softmax-in-neural-network-language-model/ said, sigma(x) in hierarchical softmax is: $\sigma([n(w,j+1)=ch(n(w,j))]\cdot v'^{T}_{n(w,j)}v_{w_I})$. Let's denote $v'^{T}_{n(w,j)}v_{w_I}$ as "simi", then sigma(simi) can be re-write as $\sigma(simi)=\frac{e^{(1-code)*simi}}{1+e^{simi}}$.

How to interpret above formular?

when code==1, $\sigma(simi)=\frac{1}{1+e^{simi}}$; when code==0, $\sigma(simi)=\frac{e^{simi}}{1+e^{simi}}$. The sum is 1. It's right.

We usually find the derivative of sigma(simi) after giving it a log(.). So, $ln\sigma(simi)=(1-code)*simi-ln(1+e^{simi})$. Its derivative over "simi" is $(1-code)-\frac{e^{simi}}{1+e^{simi}}=1-code-f$ (note that the final "f" is checked from the expTable.)

thoughts on “word2vec implementations of gradient and expTable”

1. *said:*jiangwen

   January 17, 2014 at 2:22 am

   我的理解是应该对loss function求导，也就是 1/2 * (d – f)^2, 其中f = 1 / (1 + e^x), d = code，求导结果是(code – f) * f * (1 – f).

   不知道文中的sigma(simi)从哪里来，loss function是什么形式呢

   感谢！

   REPLY

2. *said:*MEhomer

   April 17, 2015 at 11:05 am

   Hello,

   I would like to ask why is sigma(simi) = 1 / (1 + e^simi) and it is not sigma(simi) = 1 / (1 + e ^ -(simi))?

Thank you in advance for the answer,
MEhomer.

REPLY