D14：推論方法的詞向量： word2vec 的高速化

課程閱讀 本日作業 問題討論 學習心得(完成)

推論方法的詞向量： word2vec 的高速化

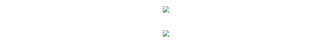
重要知識點

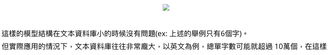
- 了解原本 word2vec 的缺點
- 了解如何透過 embedding 提高 word2vec 輸入與輸出層效率
- 了解 Hierarchical Softmax 與 Negative Sampling 如何加速 word2vec

原 Word2vec 模型缺點

先前提到的 word2vec 模型(以 window size 等於 2 的 CBOW 為例)如下圖所示，文本資料為 "I am studying natural language processing"，其中紅色的為 context(input)而藍色的字詞為 center(target/label)。在訓練的時候多個 context 共用輸入的權重，經過以下步驟

1. 兩次矩陣運算(輸入層到隱藏層，隱藏層到輸出層)
2. 利用 softmax 函數計算輸出字詞的機率分佈
3. 將輸出字詞機遂分佈與目標字詞(label)計算 Loss (一般使用Cross entropy)
4. 進行導傳遞更新模型權重





這樣的模型結構在文本資料庫小的時候沒有問題(ex: 上述的舉例只有6個字)。
但實際應用的情況下，文本資料庫往往非常龐大，以英文為例，總單字數可能就超過 10萬個，在這樣的情況下，上述的模型結構會變成輸入層為 100000 x 4 的矩陣而輸出層為 4 x 100000，且在過完 softmax 函數必須計算 100000 個字詞的分數，整個模型會變成像下圖所示：

可以發現當詞彙數量變得龐大的時候，衍生而來的問題主要會有下列 3 點

1. 輸入的字詞利用 one-hot 編碼，會佔據過多的記憶體 (需要儲存 100000 個元素)
2. 輸入與輸出層的矩陣過於龐大，因此在計算上會需要消耗較多的資源與時間
3. Softmax 需要計算所有字詞的分數(需計算自然指數 exponential 值，學員可參考softmax公式)

採用 Embedding 層

將字詞送進 word2vec 模型前，我們會將字詞轉換為 one-hot 的向量，在與輸入層的權重進行向量與矩陣的乘積運算。

從上面可以發現，再進行 one-hot 向量與矩陣乘積時大部分都是與 0 相乘積，因此大部分的計算其實都是沒有效益的。

實際上 one-hot 向量與矩陣乘積只是留由 one-hot 向量內非 0 位置對應到的矩陣列向量，因此單純取出對應的列向量，可以不需要將輸入的字詞轉換成 one-hot 格式在進行龐大的矩陣乘積，直接透過指標(index)來指定需要的列向量就可以達到目的。

- 這樣就可以解決龐大矩陣相乘以及需要大量記憶體空間儲存 one-hot 向量的問題。
- Embedding 層可以順利解決原本 word2vec中one-hot 編碼與輸入輸出層矩陣運算的問題。
- 對於最後 Softmax 計算量的問題，可以透過 Hierarchical Softmax 或 Negative Sampling 的方式來解決。

Hierarchical Softmax

Hierarchical Softmax 主要的實踐方式為霍夫曼樹(Huffman Tree)，其樹的結構如下。

霍夫曼樹中的根節點就會對應到 word2vec 中的投影層(projection layer)(上述的隱藏層)，樹的葉子結點就代表著文本資料中的所有字詞(以圖示為例，共有 V 個字)。
圖示反黑顏徑表達的即為模型輸出字詞為 W2 的路徑，途中會經過 n(W2, 1)根節點、n(W2, 2)第二個節點 - n(w2, 3)第三個節點與 W2 目標葉子節點。

霍夫曼樹中，每一個葉子節點都會有一個獨特的霍夫曼編碼(圖中的 d)，在每一個節點可以定義向左的編碼為 1、向右的為 0 (word2vec即為此種定義方式)。因此 W2 的霍夫曼編碼為: 110。

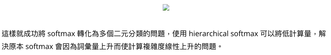
在這樣的架構下，以目標字詞為 W2 為例，問題就會變成想辦法使 P(w=w2 | w)的機率最大(即路徑機率的最大化)，其中w為跟節點的向量(hidden layer)。其中，中間的路徑每一個節點都在決定往做或往右走(二元分類)，這樣就將 softmax 轉換為多個二元分類的問題了。

這邊我們可以使用二元分類常見的激勵函數 sigmoid來計算向左或向右的機率

很簡單可以理解，若上面是被分為正樣本(向右走)的機率，那被分為負樣本(向左走)的機率即為

其中 Theta 可以理解為每個節點的向量維度與 hidden layer 輸出(根節點)相同。

所以由跟節點出發到達 W2 一共會歷經 3 次的分類



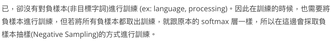
得到目標字詞 W2 就變成將下列機率最大化

這樣就成功將 softmax 轉化為多個二元分類的問題，使用 hierarchical softmax 可以將低計算量，解決原本 softmax 會因為詞彙量上升而使計算複雜度線性上升的問題。

Negative Sampling

除了上述的 Hierarchical Softmax 以為，也可以利用 Negative Sampling 的方法來解決 softmax 計算複雜度的問題。

Negative sampling 的思路與 hierarchical softmax 很相像，利用二元分類來近似多元分類的問題。同樣以 "I am studying natural language processing" 且總共有 10 萬的字詞為例，當輸入的上下文為 "am" 與 "natural" 時，的目標文字為 "studying"，因此這時只需要比對輸出層中 "studying" 的位置就好。



這樣就可以避免因為字彙詞數增加造成 softmax 計算量的問題。但是上述的方法只有訓練到正樣本而已，卻沒有對負樣本(非目標字詞進行訓練 (ex: language, processing))。因此在訓練的時候，也需要將負樣本進行訓練，但若將所有負樣本都取出訓練，就跟原本的 softmax 層一樣，所以在這邊會採取負樣本抽樣(Negative Sampling)的方式進行訓練。

在進行抽樣，我們希望高頻詞被抽中的機率比低頻詞較高，因此每個詞被抽中的機率會是

其中分母是所有字典中的字詞出現頻率的總和，分子為各字詞自身的頻率。 在 word2vec 論文中會將頻率取 0.75 次方(這個依數是可以自行嘗試變更的)

(學員可以思考次方依數會如何影響字詞的抽取機率，我們會在作業中做later的探討)

最結合正樣本(目標字詞)與負樣本(非目標字詞)，的損失函數就會如下所示

(此損失函數為binary cross entropy，學員可以參考這篇文章)

知識點回顧

在這章節我們學到了

1. 了解原本 word2vec 的缺點
2. 如何使用 embedding 層來提高 word2vec 在矩陣計算的效率
3. Hierarchical softmax 與 negative sampling 是如何解決 softmax 計算複雜的問題

延伸閱讀

二次採樣(subsampling 介紹)
網站：Medium

下一步：完成作業