[Assignment3] What is a word embedding?

Hi, I am doing A3.

In A3, there is a section about word embedding, but I am confusing because I think this part was not covered in lecture 10.

I know its solution, however, I do not know what they mean like W[x], the dimension of X and W.

Word embedding: forward

In deep learning systems, we commonly represent words using vectors. Each word of the vocabulary will be associated with a vector, and these vectors will be learned jointly with the rest of the system.

In the file `cs231n/rnn_layers.py`, implement the function `word_embedding_forward` to convert words (represented by integers) into vectors. Run the following to check your implementation. You should see an error on the order of `e-8` or less.

```
N, T, V, D = 2, 4, 5, 3

x = np.asarray([[0, 3, 1, 2], [2, 1, 0, 3]])
W = np.linspace(0, 1, num=V*D).reshape(V, D)
out, _ = word_embedding_forward(x, W)
expected_out = np.asarray([
 [[ 0.,          0.07142857,  0.14285714],
  [ 0.64285714,  0.71428571,  0.78571429],
  [ 0.21428571,  0.28571429,  0.35714286],
  [ 0.42857143,  0.5,         0.57142857]],
 [[ 0.42857143,  0.5,         0.57142857],
  [ 0.21428571,  0.28571429,  0.35714286],
  [ 0.,          0.07142857,  0.14285714],
  [ 0.64285714,  0.71428571,  0.78571429]]])

print('out error: ', rel_error(expected_out, out))
```

```
[0.         0.07142857 0.14285714] [0.64285714 0.71428571 0.78571429]
out error:  1.0000000094736443e-08
```

```
def word_embedding_forward(x, W):
    """
    Forward pass for word embeddings. We operate on minibatches of size N where
    each sequence has length T. We assume a vocabulary of V words, assigning each
    word to a vector of dimension D.

    Inputs:
    - x: Integer array of shape (N, T) giving indices of words. Each element idx
      of x muxt be in the range 0 <= idx < V.
    - W: Weight matrix of shape (V, D) giving word vectors for all words.

    Returns a tuple of:
    - out: Array of shape (N, T, D) giving word vectors for all input words.
    - cache: Values needed for the backward pass
    """
    out, cache = None, None
    ##############################################################################
    # TODO: Implement the forward pass for word embeddings.                      #
    #                                                                            #
    # HINT: This can be done in one line using NumPy's array indexing.           #
    ##############################################################################
    # *****START OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****

    out = W[x]
    cache = (x, W)

    # *****END OF YOUR CODE (DO NOT DELETE/MODIFY THIS LINE)*****
    ##############################################################################
    #                               END OF YOUR CODE                             #
    ##############################################################################
    return out, cache
```

What is word embedding and what does this code mean?

NielsRogge

4 points·15 days ago·*edited 15 days ago*

A word embedding is a vector representation of a word. For example, the word embedding of the word "cat" might look like [-0.2, 0.7, 0.55, 0.43, 0.67]. The clever idea behind word embeddings is that one tries to create a vector representation for every word in the vocabulary, such that vectors of words that are in some way similar (for example "cat" and "dog" are quite similar since they both are animals, are pets, have 4 legs, etc.) should have vectors that are close to each other when you would project them in a vector space.

How do we come up with these vectors? Actually, this is quite simple. When you would feed words into a neural network, you simply use a weight matrix of shape (vocabulary size, dimensionality of the word vectors). For example, the English vocabulary might consist of 10,000 unique words, and we choose a vector dimensionality of size 5. Then our weight matrix has shape (10,000, 5). So each of the 10,000 words will be represented by a vector that contains 5 elements. Each row in this matrix represents the word embedding for a word in the vocabulary. Each word can be identified by its row index in this matrix. So if the word "cat" is the 5698th row in this weight matrix (for example), you simply have to look up the 5698th row in the weight matrix to get the vector representation for the word "cat". You can then feed this vector into the neural network that you're training (for example an RNN that tries to predict the next word given the previous words).

The elements in this weight matrix are parameters, just as the weight matrices of an RNN for example. So you can initialize them randomly and then train them with backpropagation. After training, the rows in this matrix will be the clever word vectors.

So if you have a sentence like "I love going to the beach", you first represent this sentence by the row index of every individual word in this sentence in the weight matrix, for example x = [5, 188, 45, 8947, 20]. This means that the word "I" is the fifth row in this weight matrix, "love" is the 188th row in this weight matrix, and so on. By looking up every word vector/row in this weight matrix for each word in the sentence, you get the word vectors and you then feed them in your neural network. Note that the weight matrix containing the word vectors is almost always trained together with the neural network end-to-end.

In reality, all of this is implemented in batches of sentences rather than individual sentences. So x will be a **batch** of sentences containing the integers (row indices) of every word of every sequence. So if you have a batch of 10 sentences, and you choose to represent every sentence by 20 integers at most, x will be of shape (10, 20). If you then type W[x, :] in numpy, you will simply look up the word vectors of every sentence in the weight matrix, and the output will be of shape (10, 20, dimensionality of the word vectors).