


Applied Text Analytics & Natural Language Processing



with Dr. Mahdi Roozbahani
& Wafa Louhichi

Word2Vec (CBOW and Skip-gram)



Learning Objectives

In this course, you will learn

- A new effective embedding technique
- Two algorithms to convert a word into a vector
- Word similarity through the word vectors

Let's Quickly Review One-Hot Encoding Again

The simplest word embedding is One-hot encoding.

For simplicity, let's say we have just one document in

Apple and orange are fruit.

Using one-hot encoding (Note that I removed the common words from the corpus, such as "and")

Apple => [1,0,0,0]

orange => [0,1,0,0]

are => [0,0,0,1]

fruit => [0,0,0,1]

The Issues with One-Hot Encoding

- The size of each word vector equals the vocabulary size in our corpus. This can be huge if we have millions of words in our vocabulary
- A very long one-hot encoded vector is a waste of storage and computation
- The curse of dimensionality issue can emerge in this case of a very large vector
- If we have a new corpus, then the size of each word vector will be different, and the model we trained on the previous corpus will be useless in the case of transfer learning

Contextual Meaning of the Words

Let's go back to our example **“Apple and orange are fruit.”**

If we use one-hot encoding, can I say “apple” and “fruit” share some common features as they are both fruits?

No, because one-hot encoding is just a 0 and 1 embedding and does not consider the contextual meaning of the words. In short, there is no correlation between words that have similar meanings or usage.

What Do We Want to Achieve from Word Embedding?

“Apple and orange are fruit.”

Can we come up with a word embedding that can capture a numerical similarity value?

Similarity value of (apple and orange) == Similarity value of (orange and apple)

Similarity value of (apple and orange) > Similarity value of (apple and are)

First Algorithm: Continuous Bag of Words Model (CBOW)

This algorithm will use neural networks to learn the underlying representation of words

One big caveat!!

A Neural networks model is a supervised algorithm, and it needs labels

We need to develop a way to synthesize the labels from our corpus

Neighboring Words for CBOW (Label Creation)

“Apple and orange are fruit.”

We are going to say, given the neighbors of a word can we predict the center word?

Apple and ----- are fruit

Given the context or neighbors of the blank, can I predict the blank by a window size (the window size is the hyper-parameter)?

For simplicity, a window size of one for blank will be (Apple, center word, are).

Note: We removed the common words (i.e. and)

In fact, we want to find the $P(\text{orange}|\text{context})$, and we need to maximize this probability

Neighboring Words for CBOW (Label Creation)

“Apple and orange are fruit.”

We are going to say, given the neighbors of a word can we predict the center word?

Apple and ----- are fruit

Given the context or neighbors of the blank, can I predict the blank by a window size (the window size is the hyper-parameter)?

For simplicity, a window size of one for blank will be (Apple, center word, are).

Note: We removed the common words (i.e. and)

In fact, we want to find the $P(\text{orange}|\text{context})$, and we need to maximize this probability

Embedding Every Single Word in the Corpus Using its Context

“Apple and ----- are fruit.”

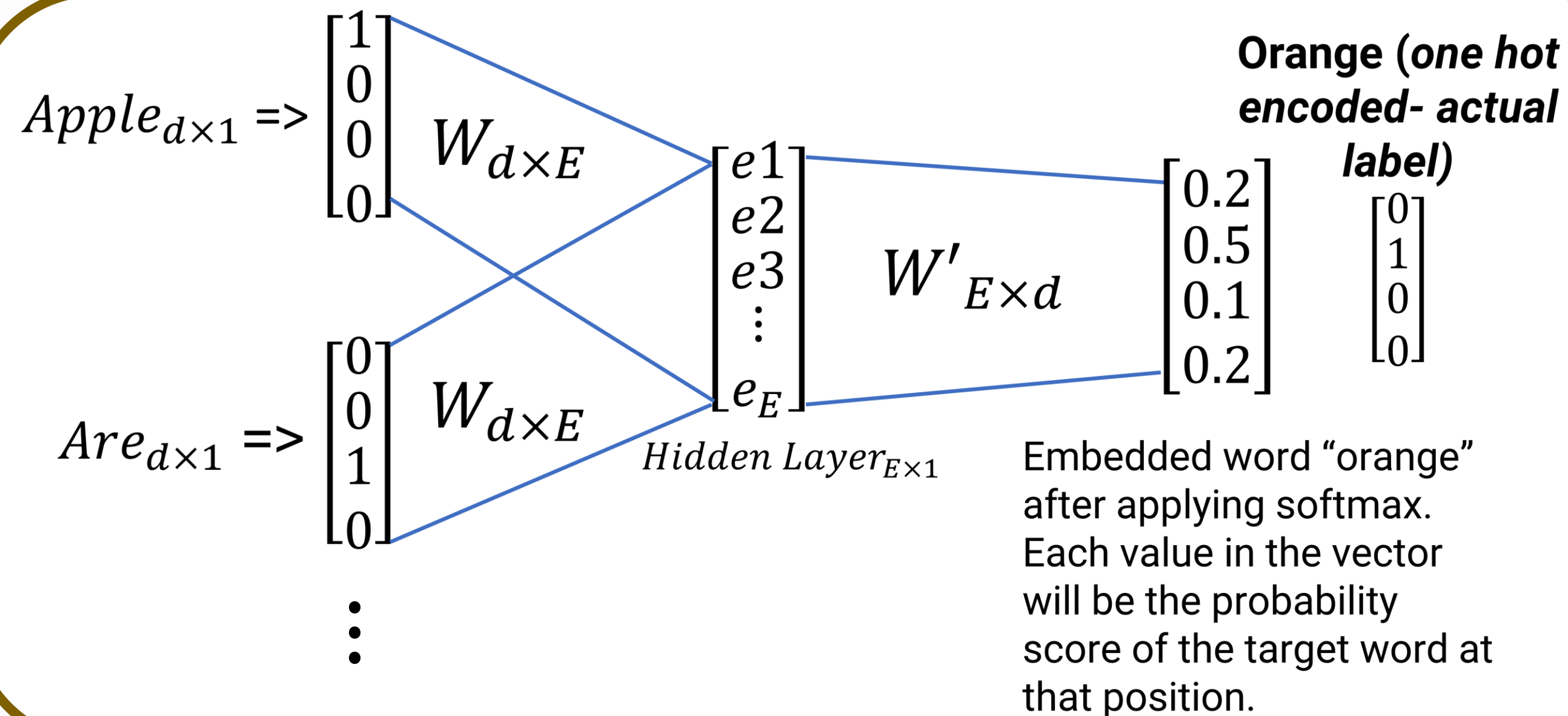
We are going to find the embedding representation of the word “**orange**”

First, all the words in the vocabulary need to be encoded using one-hot encoding. Each word will have d dimensions (the size of the vocabulary).

$$\text{Apple} \Rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{orange} \Rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{are} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{fruit} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

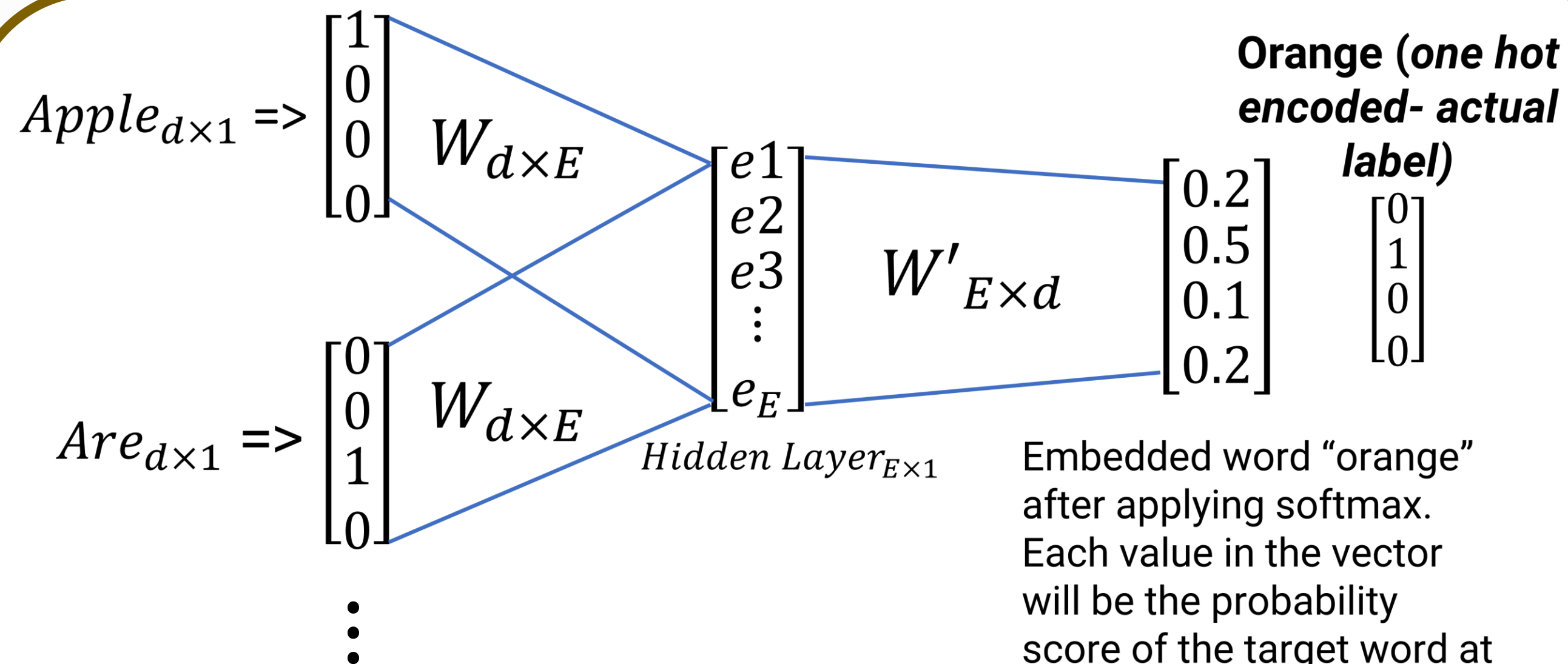
Using a Window Size of One

[Apple , Center Word , Are]



Using a Window Size of One

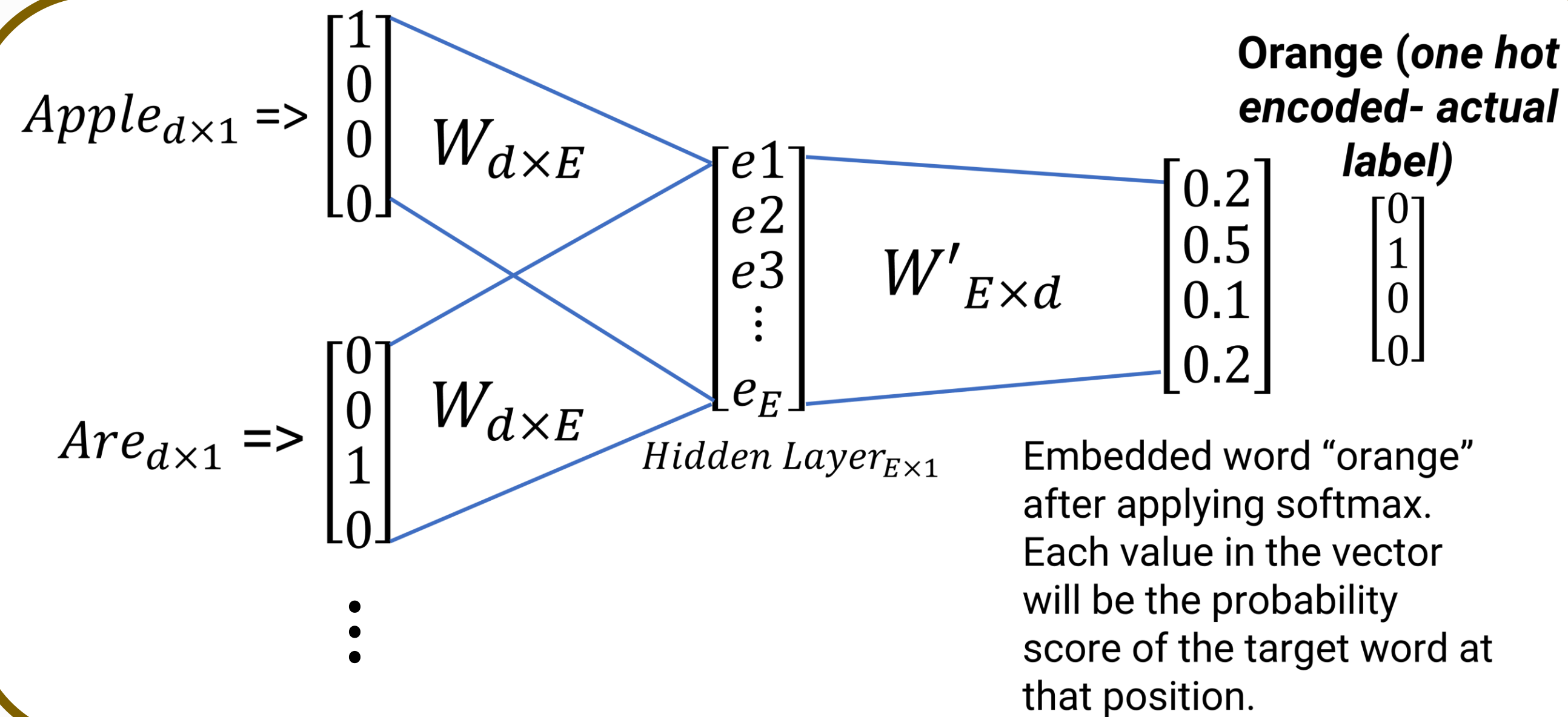
[Apple , Center Word , Are]



Embedded word "orange" after applying softmax. Each value in the vector will be the probability score of the target word at that position.

Using a Window Size of One

[Apple , Center Word , Are]



Second Algorithm: Skip-Gram Model

“Apple and orange are fruit.”

We are going to say, given a center word, what could be the context (neighbors) of the center word? Opposite of CBOW

----- orange --- fruit

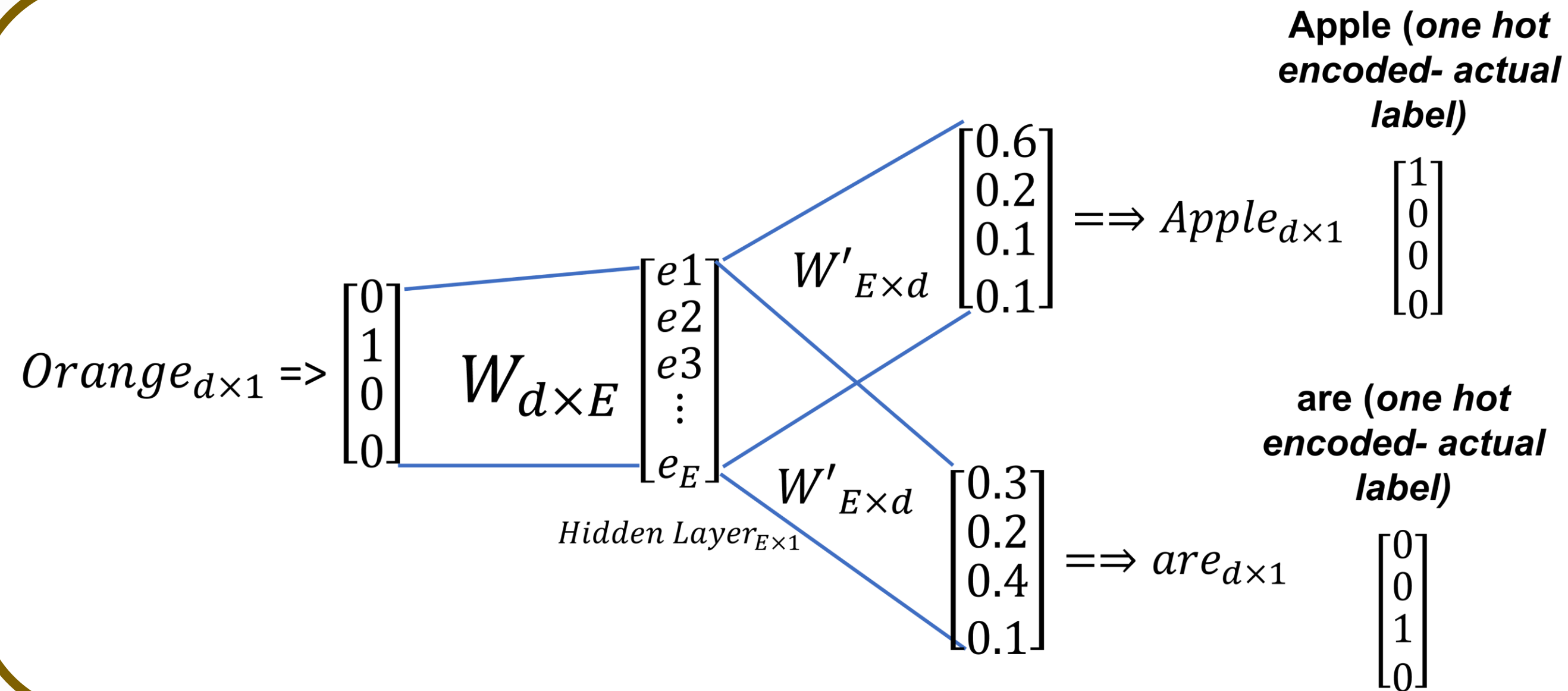
Given the center word (orange), can I predict the blanks by a window size (the window size is the hyper-parameter)?

In fact, we want to find the $P(\text{context}|\text{orange})$, and we need to maximize this probability.



Using a Window Size of One

[Apple -- Center Word -- Are]



CBOW or Skip-Gram Model?

CBOW vs Skip-gram

1. CBOW Learn better syntactic relationships between words while Skip-gram is better in capturing better semantic relationships (CBOW would provide cat and cats as similar while skip-gram will provide cat and dog as similar)
2. CBOW is trained to predict (maximize the probability) a single word from a fixed window size of context words, while Skip-gram does the opposite and strives to predict several context words from a single input word
3. CBOW is much faster to train than Skip-gram

Summary

- We learned about Word2Vec
- We learned about two algorithms CBOW and Skip-gram
- These embeddings capture the contextual meaning of words