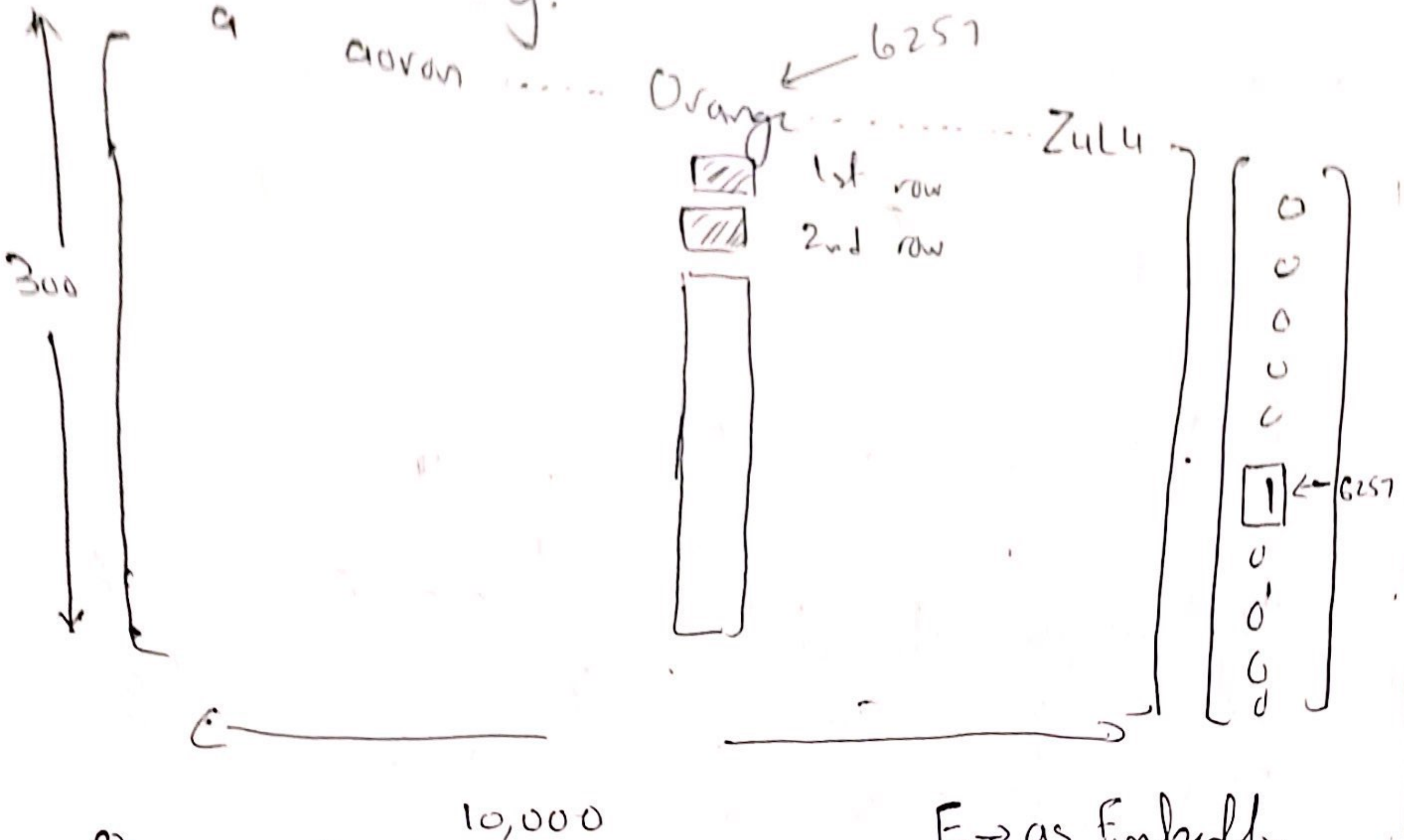
How do you learn embeddigs

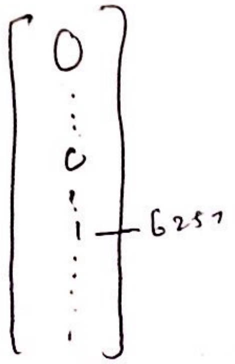
Embedding Matrix:

Let's say we are working on a 10,000 word vocabulary.



E → as Embedding Matrix

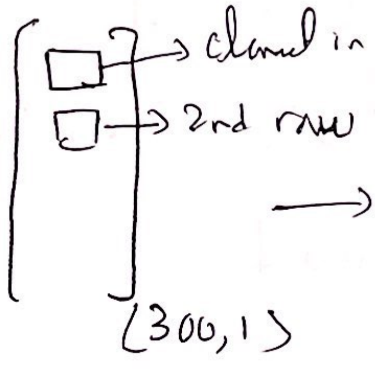
O_{6257} → 1-hot vector



e_w = Embedding for word_w

$E \cdot O_{6257} =$

300x 10,000 ; dot product ; 10,000 X1



→ cloned in Orange Column

→ 2nd row

$= e_{6257}$ 300x1 Vector

→ 300-dimension Vector corresponding to word Orange

More generally E times O

$E_j \cdot O_j = e_j$ embedding for word j

goal is to learn E_j

1- Initialize randomly and learn all parameters of this 300 by 10,000 dimension matrix.

A Language model (LM) is the use of various statistical and probablistic techniques to determining the probability of a given sequence of words occuring in a sentence.

Lets assume we are building an LM. using NN

Task:

I want a glass of Orange ______

4343 9665 1 3852 6163 6257

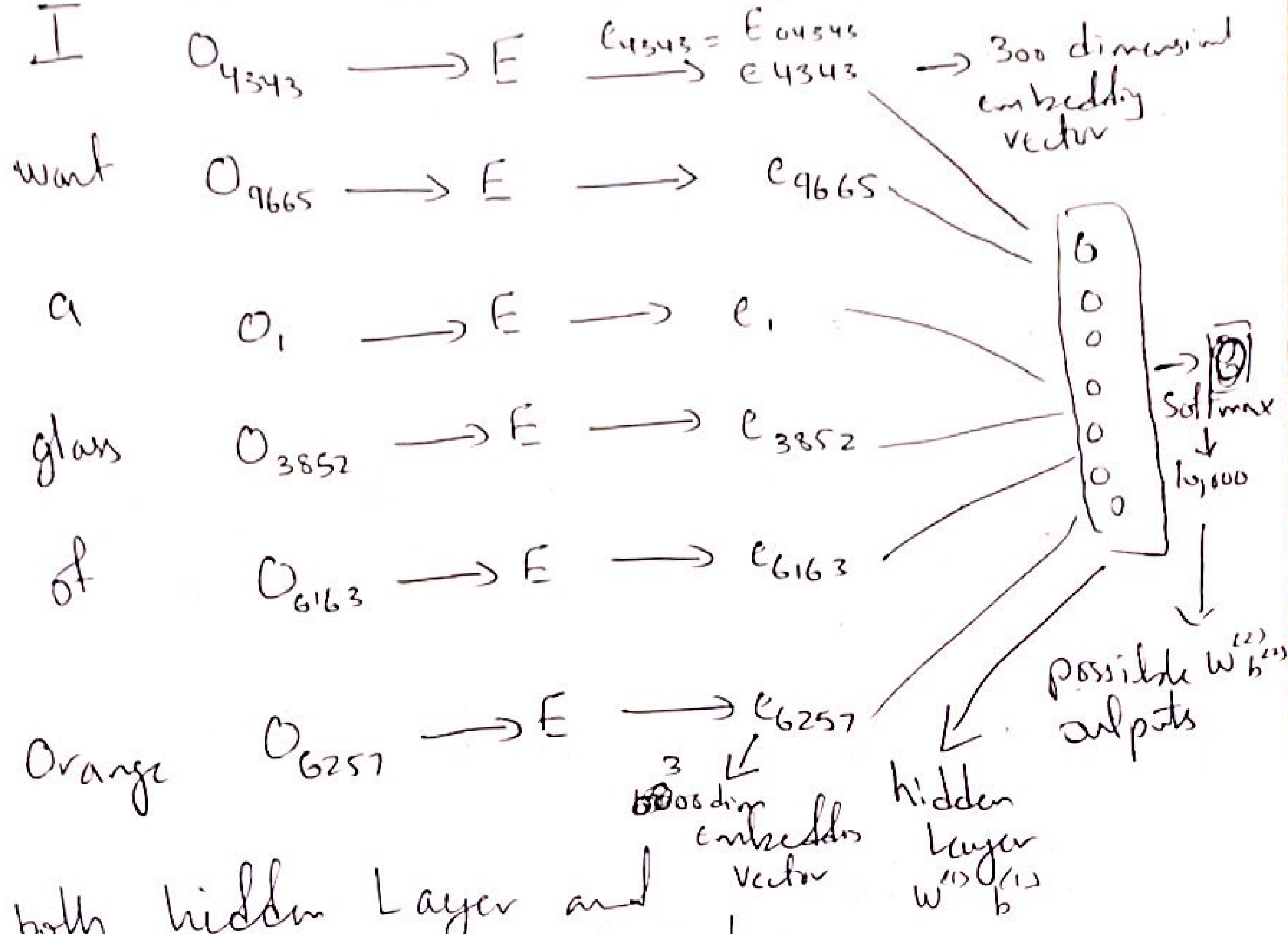
→ index in vocabulary.

predict the next word

Predicting the next word in the sequence is a reasonable way of learning set of embeddings.

I want a glass of orange ______

4343 9665 1 3852 6163 6257

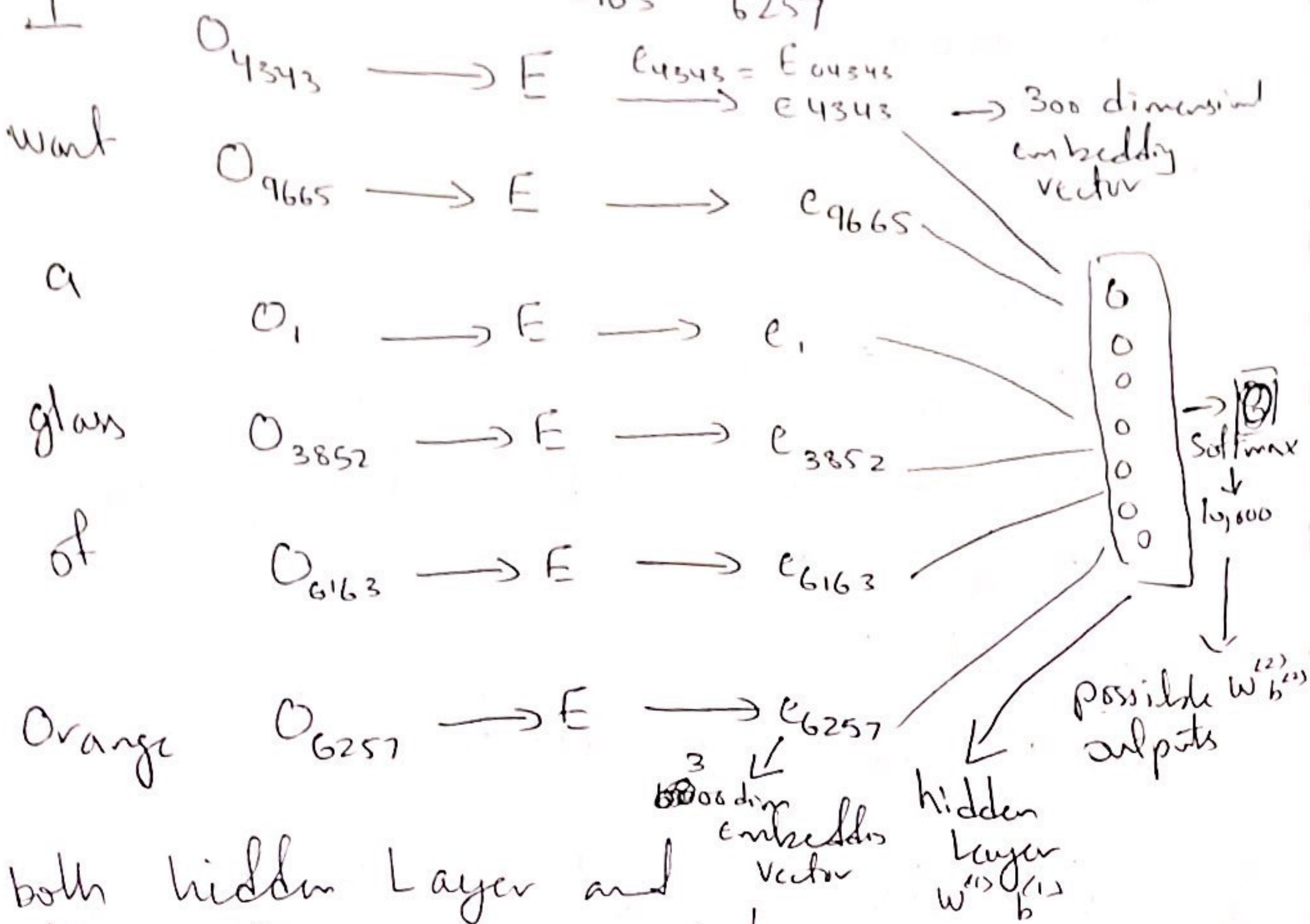

I O_{4343} → E $\xrightarrow{e_{4343} = E\,O_{4343}}$ e_{4343} → 300 dimensional embedding vector
want O_{9665} → E → e_{9665}
a O_1 → E → e_1
glass O_{3852} → E → e_{3852}
of O_{6163} → E → e_{6163}
Orange O_{6257} → E → e_{6257}
300 dim embeddin vector
hidden Layer $W^{(1)}$ $b^{(1)}$
softmax
10,000
possible $W^{(2)}$ $b^{(2)}$ outputs

both hidden Layer and the softmax layer will have their own params.

1800 input layer
6 x 300 stacking them together.

Predicting the next word in the sequence is a reasonable way of learning set of embeddings.

I want a glass of Orange ______
4343 9665 1 3852 6163 6257



I $O_{4343} \rightarrow E \xrightarrow{e_{4343} = E\,O_{4343}} e_{4343}$
→ 300 dimensional embedding vector
want $O_{9665} \rightarrow E \rightarrow e_{9665}$
a $O_1 \rightarrow E \rightarrow e_1$
glass $O_{3852} \rightarrow E \rightarrow e_{3852}$
of $O_{6163} \rightarrow E \rightarrow e_{6163}$
Orange $O_{6257} \rightarrow E \rightarrow e_{6257}$
300 dim embedding vector
hidden Layer $W^{(1)}$ $b^{(1)}$
Softmax
10,000
possible outputs
$W^{(2)}$ $b^{(2)}$

both hidden Layer and the softmax layer will have their own params.

1800 input layer
6 x 300 stacking them together.

have a fixed historical window

a glass of orange ——

predict the next word given previous four words.

In such a case NN will input $4 \times 300 = 1200$ dimensional input feature vector

Using a fixed window means we can deal with arbitrary long sentences because input size is fixed.

the parameters matrix 'E', same matrix for all the words.

$w^{<1>}, b^{<1>}$ and $w^{<2>} b^{<2>}$ are also parameters of the algorithm.

Use backpropagation to perform gradient descent to predict, given 4 words in the sequence what will be the next word in the text.

This algorithm will perform reasonably well.

Example: Orange juice
Apple juice

Algo will give similar weights to orange and Apple.

if we have 300 features the algo will ~~fit better or~~ find it best if fruits will end up with similar feature vectors.

Other Algorithm Complex Example

I want a glass of orange juice to go along with my cereal.

context — target

a) Context: Last 4 words

b) 4 words left and right

a glass of orange ___?___ to go along with

c) Last 1 word

orange ?

NN with just 1 word.

d) take nearby 1 word

glass ?

this is an idea of skip gram

for Language model: use the last few words. as context

to learn word embedding. use all of these contexts.

Two ways to implent word2vec, CBOW. and skip-gram

CBOW: a window around some target word and consider the words around it. (context)

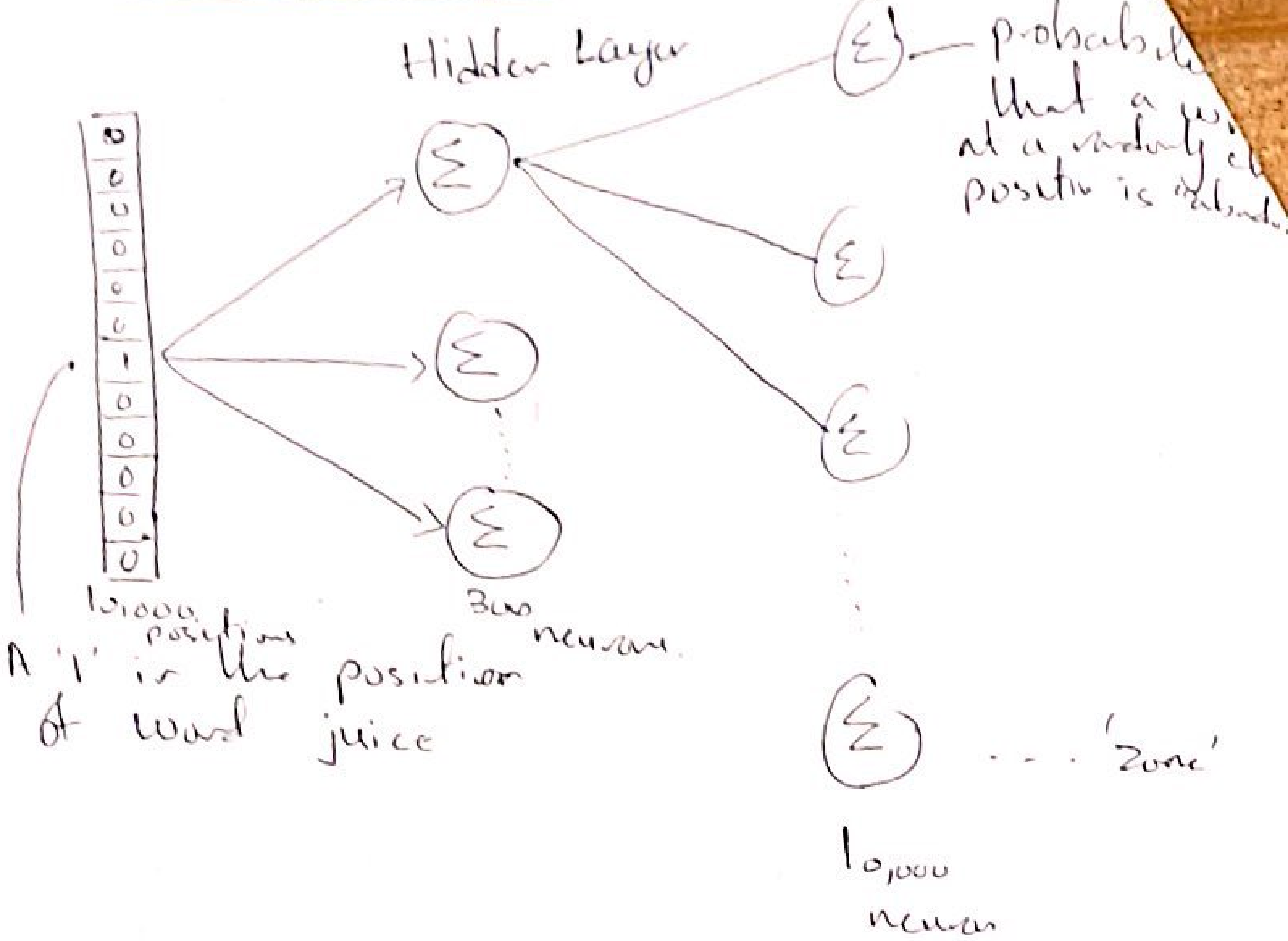
We supply context words and use it to predict the target word.

skip-gram: We have a target word and we try to predict the words that are in the window. around that word. i.e. predict the context

The input words are passed in as one-hot encoded vectors.

then we will send it to a hidden layer then to a softmax layer

The weights of the hidden layer represents word embeddings


Hidden Layer
probability that a wo at a randomly ch position is 'abandon'
0
0
0
0
0
0
1
0
0
0
0
0
10,000 positions
300 neurons
A '1' in the position of word juice
'zone'
10,000 neurons

The embedding matrix has a size of number of words by the number of neurons in the hidden layer. So for 10,000 words and 300 hidden units, the matrix will have 10,000x300

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{pmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{pmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

for the fourth word

Word2vec Model

skip-gram

I want a glass of orange juice to go along with my cereal.

Context	Target
Orange	