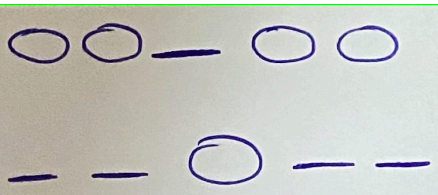
word2vec → skip-Gram

→ CBOW



unsupervised classifier

Skip Gram with negative sample (SGNS)

$P(+|t,c)$

input of classifier → t, c

output of classifier → $P(+|t,c)$, $P(-|t,c)$

→ return the probability that c is a real context word! (True or False)

compute the probability P? intuition of skip-Gram is to base this probability on similarity.

(A word is likely to occur near the target if its embedding is similar to target embedding)

→ Similarity in vectors (two vectors) are similar if they have a high dot product

So ⇒ similarity $(t,c) \approx t \cdot c$

problem → similarity of $\vec{t}$ and $\vec{c}$ is not Probability

→ $-\infty$ to $+\infty$

solution → sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$

(2) So probability that word **c** is a real context

True probability $\Rightarrow P(+|t,c) = \frac{1}{1+e^{-t.c}}$

note $\Rightarrow$ this equation gives us the probability for one word. But we need to shake it over all context window.

* Assumption : All context words are independent. Allow us to just multiply their probabilities.

$$P(+|t,c) \longrightarrow P(+|t, c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t.c_i}}$$

* $k = 2 * c$
* $c$ = context window size

---

Learning skip-Grams:

initial embeddeling process with some pos and Neg samples.

... lemon, a [tablespoon of apricot jam, a] pinch...

(+)

| t | c |
|---|---|
| apricot | of |
| apricot | tablespoon |
| apricot | jam |
| apricot | a |

(−)

| t | c | t | c |
|---|---|---|---|
| apricot | my | apricot | seven |
| ⋮ | ⋮ | ⋮ | ⋮ |

③ * training a Binary classifier with pos and Neg samples.

* skip-Gram uses more Neg examples than pos

How many ⇒ K → in our example = 2

so → for each pos example t,c , we have 2 neg.

Add noise → should we add every words in our neg samples? NO

if we sample based on unweighted freq, P(w)

so → p("the"), p("aardvark") are same. which is actually not a right manner.

we need a weight → $\alpha = 0.75$

$P^{3/4}(w) \Rightarrow P_\alpha(w) > P(w)$

→ Because it gives rare words

---

Finally → the Goal of classifier

$$L(\theta) = \sum_{(t,c)\in +} \log(+|t,c) + \sum_{(t,c)\in -} \log(-|t,c)$$

objective

maximize

minimize

we can use SGD to train this objective.

↳ iteratively modifying the parameters to max obje

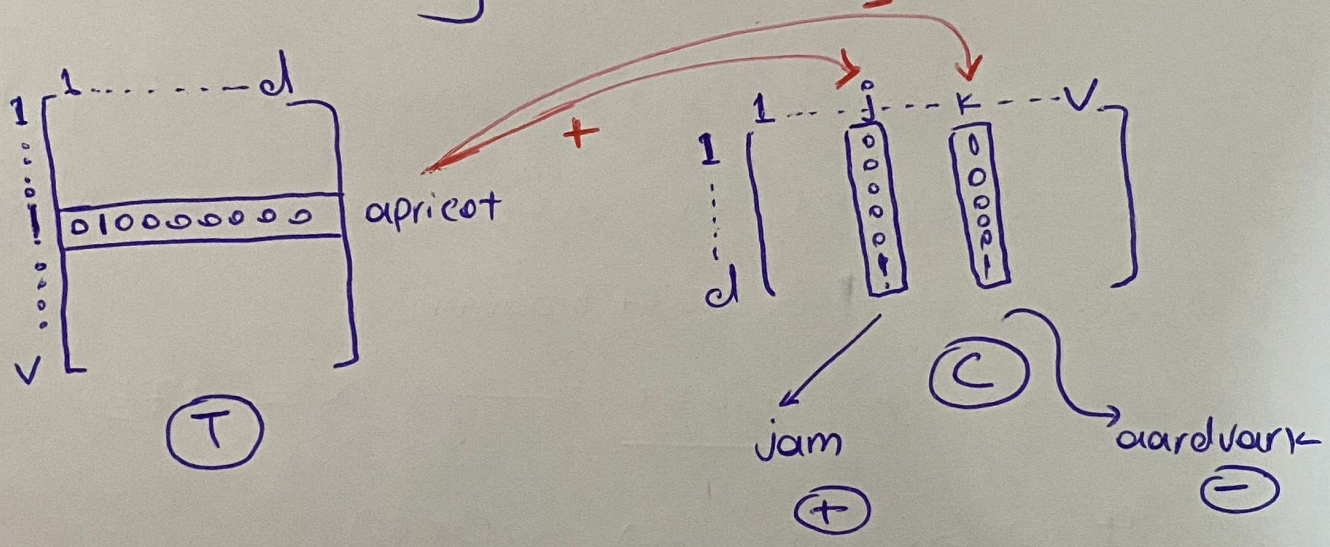
(embedding for each target word t and each context word)

---

embedding matrices

target embedding $t \Rightarrow 1 \times d$

context embedding $c \Rightarrow d \times 1$




---