NLP - Section 6

Word Vectors

Embeddings

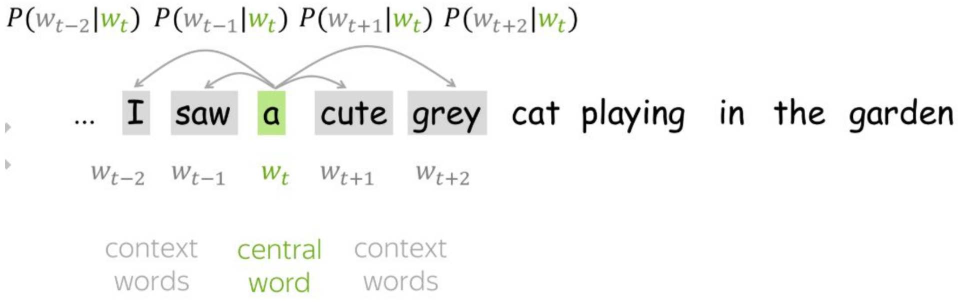
- An embedding is a numerical representation of a word.
- The representation of the word doesn’t account for the context.
- Words that occur in the same context have close numerical representation
- Two types:

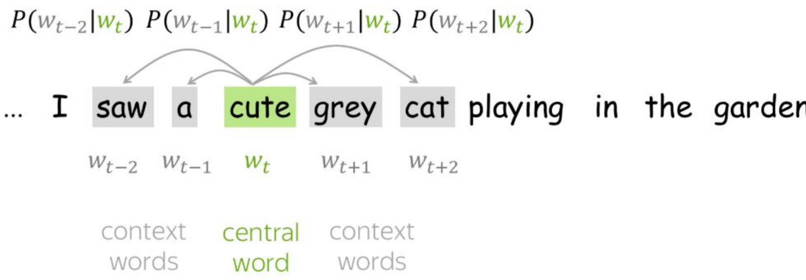
  A. Word Embeddings: no context

  B. Contextual Embeddings: with context

Word2Vec

- There are various techniques for word embeddings, one of which is Word2Vec
- Word2Vec is based on continuous skip-gram architecture where it predicts the context words given the current words.





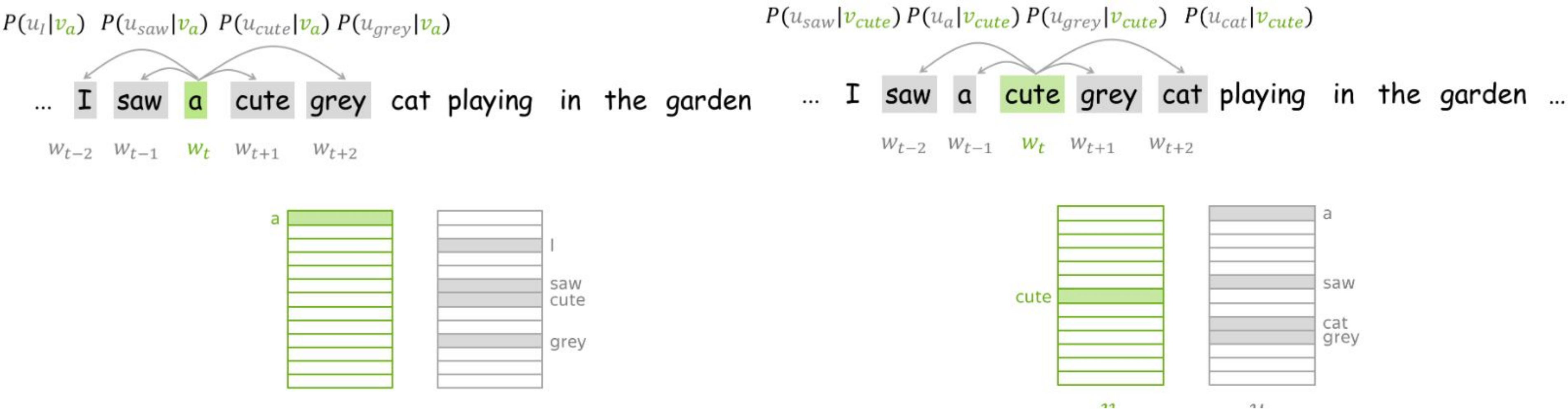
Word2Vec Cont.

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^{T} \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} | w_t, \theta),$$

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} \log P(w_{t+j} | w_t, \theta)$$

agrees with our plan above ⟼ go over text with a sliding window compute probability of the context word given the central

Skip Gram



Source: https://lena-voita.github.io/nlp_course/word_embeddings.html

Word2Vec Calculations

how to calculate this

$$\text{Loss} = J(\theta) = -\frac{1}{T}\log L(\theta) = -\frac{1}{T}\sum_{t=1}\sum_{\substack{-m\le j\le m,\\ j\ne 0}} \boxed{\log P(w_{t+j}|w_t,\theta)}$$

agrees with our plan above ⟼ go over text

with a sliding window

compute probability of the context word given the central

- We use softmax function

$$P(o|c) = \frac{\exp(\boxed{u_o^T v_c})}{\sum_{w\in V}\exp(u_w^T v_c)}$$

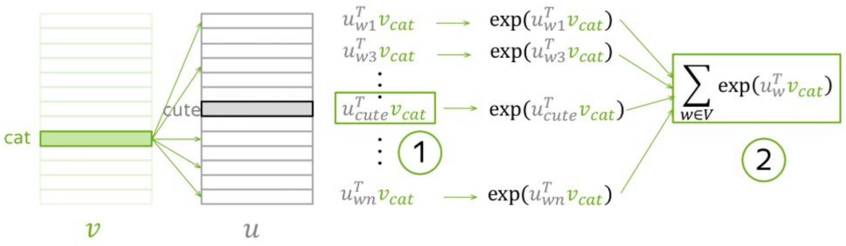
Dot product: measures similarity of o and c
Larger dot product = larger probability

Normalize over entire vocabulary
to get probability distribution

Gradient Descent Training

$$J_{t,j}(\theta) = -\log P(cute|cat) = -\log \frac{\exp u_{cute}^T v_{cat}}{\sum_{w\in Voc} \exp u_w^T v_{cat}} = -u_{cute}^T v_{cat} + \log \sum_{w\in Voc} \exp u_w^T v_{cat}.$$

1. Take dot product of v_{cat} with all u
2. exp
3. sum all



4. get loss (for this one step)

$$J_{t,j}(\theta) = \underbrace{-u_{cute}^T v_{cat}}_{1} + \underbrace{\log \sum_{w\in V} \exp(u_w^T v_{cat})}_{2}$$

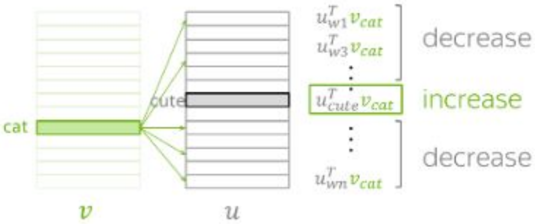
5. evaluate the gradient, make an update

$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall\, w \in V$$

Negative Sampling

Dot product of v_{cat}:
- with u_{cute} - increase,
- with **all other** u - decrease

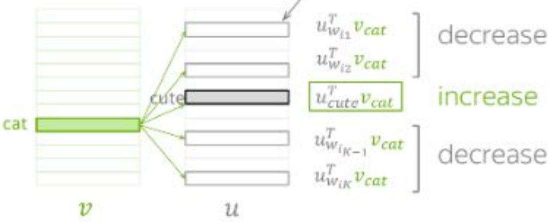


Parameters to be updated:
- v_{cat}
- u_w for all w in the vocabulary

|V| + 1 vectors

Dot product of v_{cat}:
- with u_{cute} - increase,
- with **a subset of other** u - decrease

Negative samples: randomly selected K words



Parameters to be updated:
- v_{cat}
- u_{cute} and u_w for w in K negative examples

K + 2 vectors

Negative Sampling Cont.

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log \sigma(-u_w^T v_{cat}),$$

Note that $\sigma(-x) = \frac{1}{1+e^x} = \frac{1 \cdot e^{-x}}{(1+e^x) \cdot e^{-x}} = \frac{e^{-x}}{1+e^{-x}} = 1 - \frac{1}{1+e^x} = 1 - \sigma(x)$. Then the loss can also be written as:

$$J_{t,j}(\theta) = -\log \sigma(u_{cute}^T v_{cat}) - \sum_{w \in \{w_{i_1}, \ldots, w_{i_K}\}} \log(1 - \sigma(u_w^T v_{cat})).$$

Training

1. We initialize the matrices for the embeddings and the context.
   Both have the same (vocab_size, embedding_size)
2. We create a dataset with 0 and 1, 0 is not in context and 1 is in context of the center word
3. We then train the model using gradient descent to predict if the word is in context or not of the center, this spans the entire corpus.

Code

- Word2Vec Implementation from Scratch