Module 10.4: Continuous bag of words model

- The methods that we have seen so far are called **count based models** because they use the co-occurrence counts of words

- The methods that we have seen so far are called **count based models** because they use the co-occurrence counts of words
- We will now see methods which directly **learn** word representations (these are called **(direct) prediction based models**)

The story ahead ...

- Continuous bag of words model

The story ahead ...

- Continuous bag of words model
- Skip gram model with negative sampling (the famous word2vec)

The story ahead ...

- Continuous bag of words model
- Skip gram model with negative sampling (the famous word2vec)
- GloVe word embeddings

The story ahead ...

- Continuous bag of words model
- Skip gram model with negative sampling (the famous word2vec)
- GloVe word embeddings
- Evaluating word embeddings

The story ahead ...

- Continuous bag of words model
- Skip gram model with negative sampling (the famous word2vec)
- GloVe word embeddings
- Evaluating word embeddings
- Good old SVD does just fine!!

- **Consider this Task:** Predict n-th word given previous n-1 words

- **Consider this Task:** Predict n-th word given previous n-1 words
- **Example:** he sat on a chair

Sometime in the 21st century, Joseph Cooper, a widowed former engineer and former NASA pilot, runs a farm with his father-in-law Donald, son Tom, and daughter Murphy, It is post-truth society (Cooper is reprimanded for telling Murphy that the Apollo missions did indeed happen) and a series of crop blights threatens humanity's survival. Murphy believes her bedroom is haunted by a poltergeist. When a pattern is created out of dust on the floor, Cooper realizes that gravity is behind its formation, not a "ghost". He interprets the pattern as a set of geographic coordinates formed into binary code. Cooper and Murphy follow the coordinates to a secret NASA facility, where they are met by Cooper's former professor, Dr. Brand.

Some sample 4 word windows from a corpus

- **Consider this Task:** Predict n-th word given previous n-1 words
- **Example:** he sat on a chair
- **Training data:** All n-word windows in your corpus

Sometime in the 21st century, Joseph Cooper, a widowed former engineer and former NASA pilot, runs a farm with his father-in-law Donald, son Tom, and daughter Murphy, It is post-truth society (Cooper is reprimanded for telling Murphy that the Apollo missions did indeed happen) and a series of crop blights threatens humanity's survival. Murphy believes her bedroom is haunted by a poltergeist. When a pattern is created out of dust on the floor, Cooper realizes that gravity is behind its formation, not a "ghost". He interprets the pattern as a set of geographic coordinates formed into binary code. Cooper and Murphy follow the coordinates to a secret NASA facility, where they are met by Cooper's former professor, Dr. Brand.

Some sample 4 word windows from a corpus

- **Consider this Task:** Predict n-th word given previous n-1 words
- **Example:** he sat on a chair
- **Training data:** All n-word windows in your corpus
- Training data for this task is easily available (take all n word windows from the whole of wikipedia)

Sometime in the 21st century, Joseph Cooper, a widowed former engineer and former NASA pilot, runs a farm with his father-in-law Donald, son Tom, and daughter Murphy, It is post-truth society (Cooper is reprimanded for telling Murphy that the Apollo missions did indeed happen) and a series of crop blights threatens humanity's survival. Murphy believes her bedroom is haunted by a poltergeist. When a pattern is created out of dust on the floor, Cooper realizes that gravity is behind its formation, not a "ghost". He interprets the pattern as a set of geographic coordinates formed into binary code. Cooper and Murphy follow the coordinates to a secret NASA facility, where they are met by Cooper's former professor, Dr. Brand.

Some sample 4 word windows from a corpus

- **Consider this Task:** Predict n-th word given previous n-1 words
- **Example:** he sat on a chair
- **Training data:** All n-word windows in your corpus
- Training data for this task is easily available (take all n word windows from the whole of wikipedia)
- For ease of illustration, we will first focus on the case when $n = 2$ (*i.e.*, predict second word based on first word)

We will now try to answer these two questions:

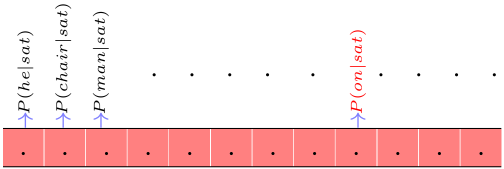
We will now try to answer these two questions:

- How do you model this task?

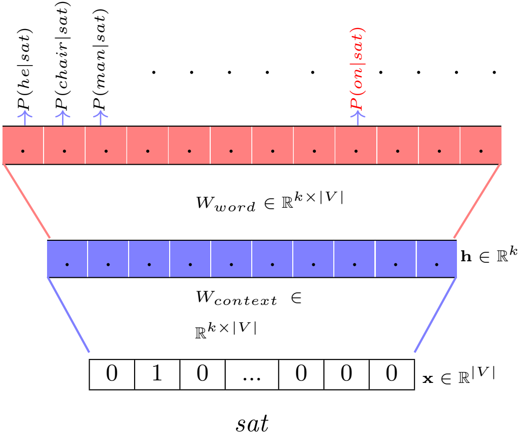
We will now try to answer these two questions:

- How do you model this task?
- What is the connection between this task and learning word representations?

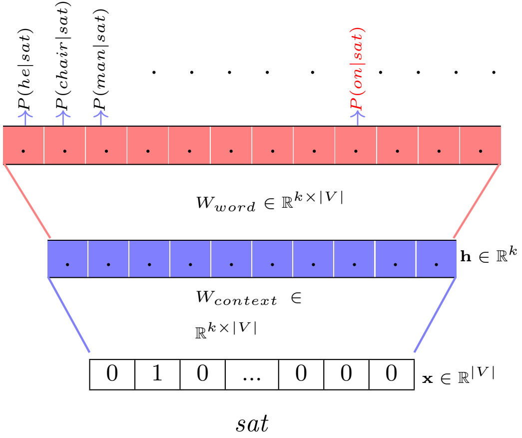
- We will model this problem using a feedforward neural network



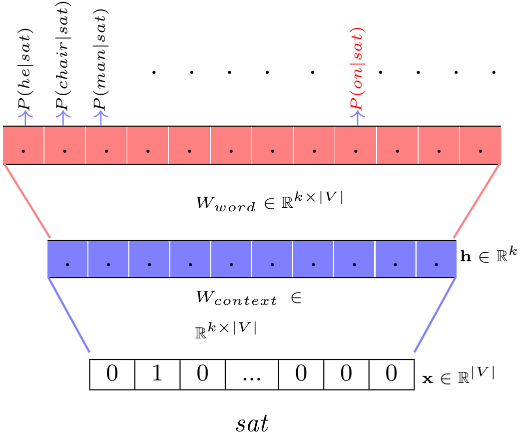
- We will model this problem using a feedforward neural network
- **Input:** One-hot representation of the **context word**



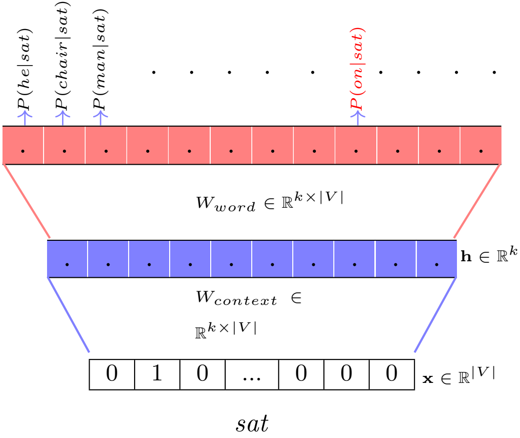
- We will model this problem using a feedforward neural network
- **Input:** One-hot representation of the **context word**
- **Output:** There are $|V|$ words (classes) possible and we want to predict a probability distribution over these $|V|$ classes (multi-class classification problem)



- We will model this problem using a feedforward neural network
- **Input:** One-hot representation of the **context word**
- **Output:** There are $|V|$ words (classes) possible and we want to predict a probability distribution over these $|V|$ classes (multi-class classification problem)
- **Parameters:** $\mathbf{W}_{context} \in \mathbb{R}^{k \times |V|}$ and $\mathbf{W}_{word} \in \mathbb{R}^{k \times |V|}$ (we are assuming that the set of **words** and **context** words is the same: each of size $|V|$)

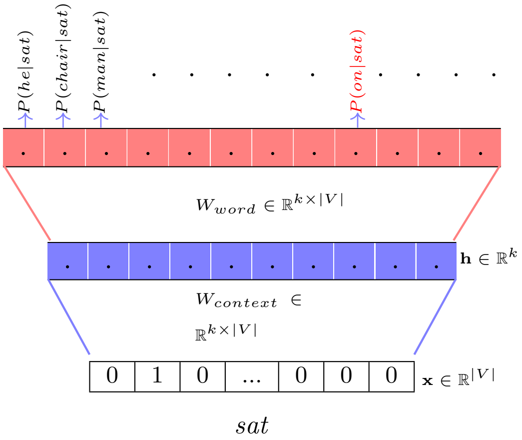


- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector



- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector
- It is simply the i-th column of $\mathbf{W}_{context}$

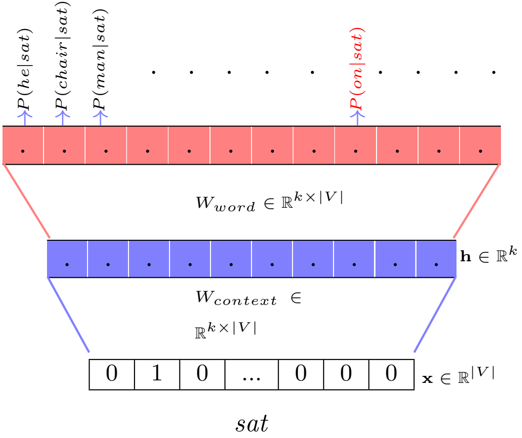
$$\begin{bmatrix} -1 & 0.5 & 2 \\ 3 & -1 & -2 \\ -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.7 \end{bmatrix}$$



- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector
- It is simply the i-th column of $\mathbf{W}_{context}$

$$\begin{bmatrix} -1 & 0.5 & 2 \\ 3 & -1 & -2 \\ -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.7 \end{bmatrix}$$

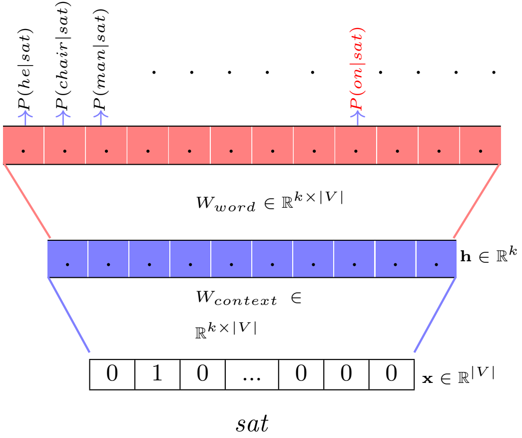
- So when the i^{th} word is present the i^{th} element in the one hot vector is ON and the i^{th} column of $\mathbf{W}_{context}$ gets selected



- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector
- It is simply the i-th column of $\mathbf{W}_{context}$

$$\begin{bmatrix} -1 & 0.5 & 2 \\ 3 & -1 & -2 \\ -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.7 \end{bmatrix}$$

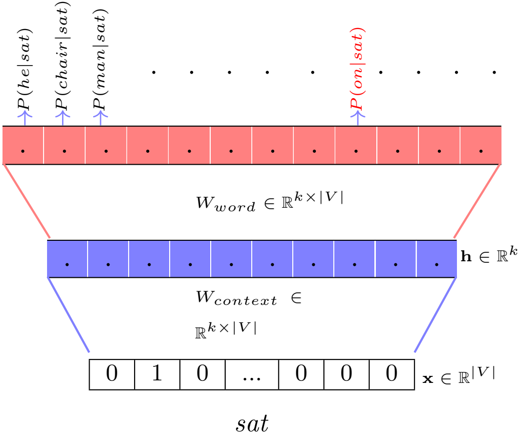
- So when the i^{th} word is present the i^{th} element in the one hot vector is ON and the i^{th} column of $\mathbf{W}_{context}$ gets selected
- In other words, there is a one-to-one correspondence between the words and the column of $\mathbf{W}_{context}$



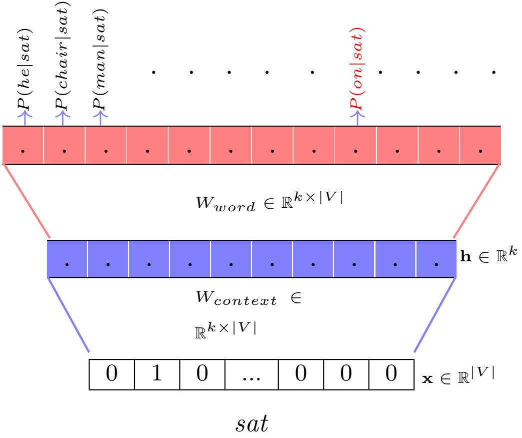
- What is the product $\mathbf{W}_{context}\mathbf{x}$ given that $\mathbf{x}$ is a one hot vector
- It is simply the i-th column of $\mathbf{W}_{context}$

$$\begin{bmatrix} -1 & 0.5 & 2 \\ 3 & -1 & -2 \\ -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.7 \end{bmatrix}$$

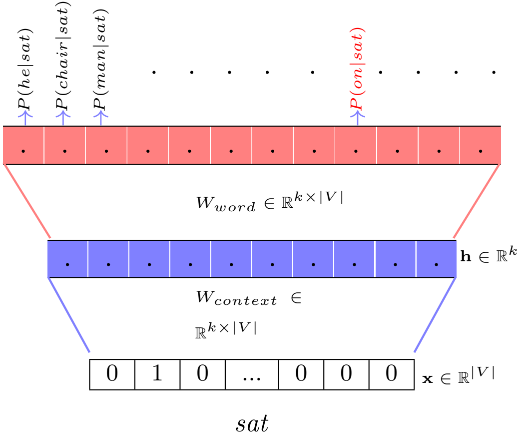
- So when the i^{th} word is present the i^{th} element in the one hot vector is ON and the i^{th} column of $\mathbf{W}_{context}$ gets selected
- In other words, there is a one-to-one correspondence between the words and the column of $\mathbf{W}_{context}$
- More specifically, we can treat the i-th column of $\mathbf{W}_{context}$ as the representation of context i



- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function?

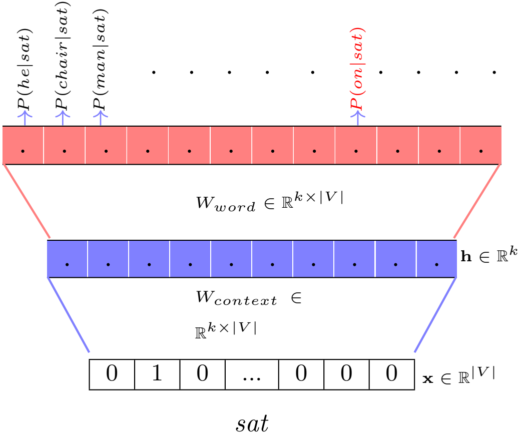


- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)



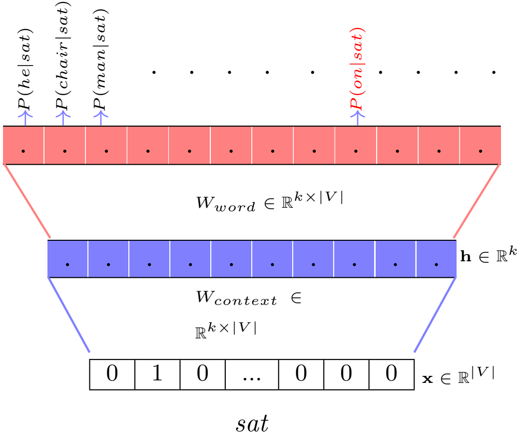
$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)



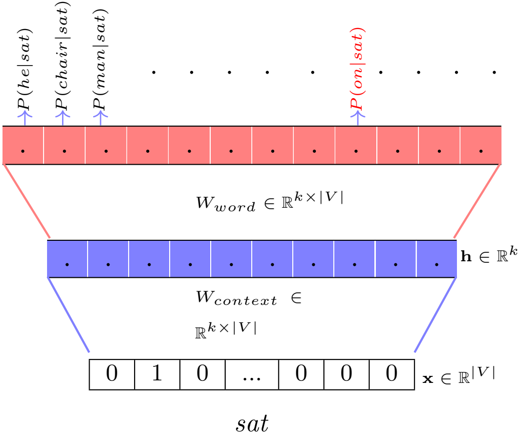
$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$



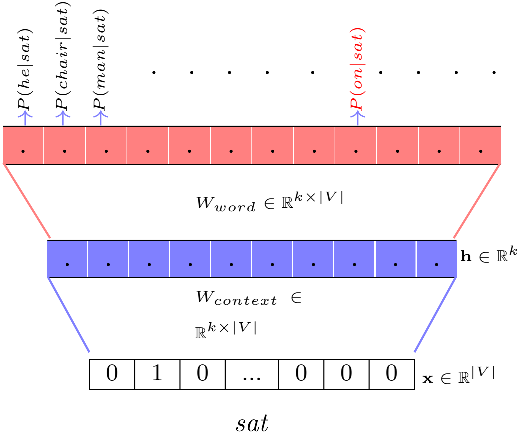
$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$
- $P(word = i|sat)$ thus depends on the i^{th} column of $\mathbf{W}_{word}$



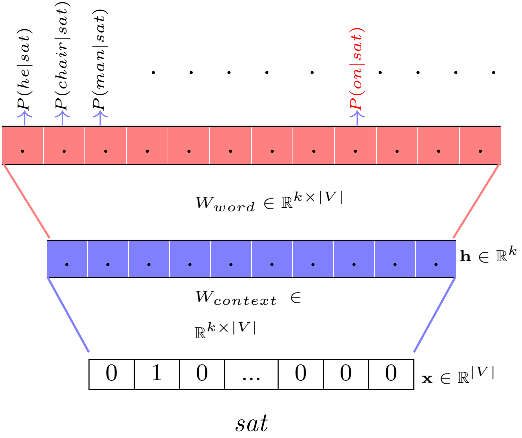
$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$
- $P(word = i|sat)$ thus depends on the i^{th} column of $\mathbf{W}_{word}$
- We thus treat the i-th column of $\mathbf{W}_{word}$ as the representation of word i



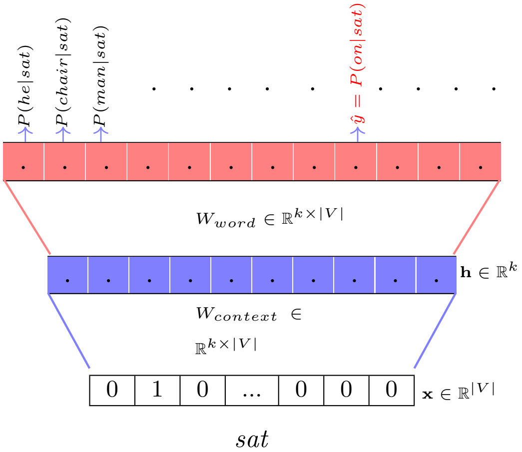
$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$
- $P(word = i|sat)$ thus depends on the i^{th} column of $\mathbf{W}_{word}$
- We thus treat the i-th column of $\mathbf{W}_{word}$ as the representation of word i
- Hope you see an analogy with SVD! (there we had a different way of learning $\mathbf{W}_{context}$ and $\mathbf{W}_{word}$ but we saw that the i^{th} column of $\mathbf{W}_{word}$ corresponded to the representation of the i^{th} word)

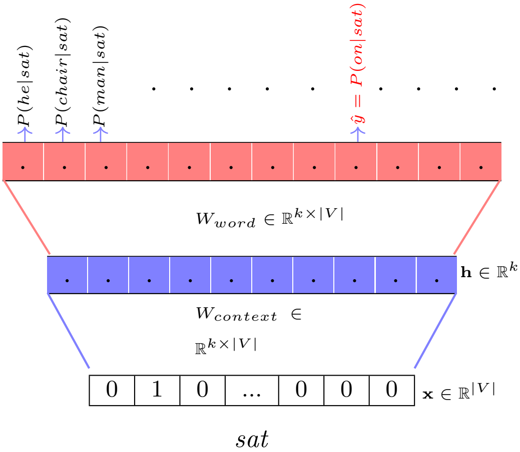


$$P(on|sat) = \frac{e^{(\mathbf{W}_{word}h)[i]}}{\sum_j e^{(\mathbf{W}_{word}h)[j]}}$$

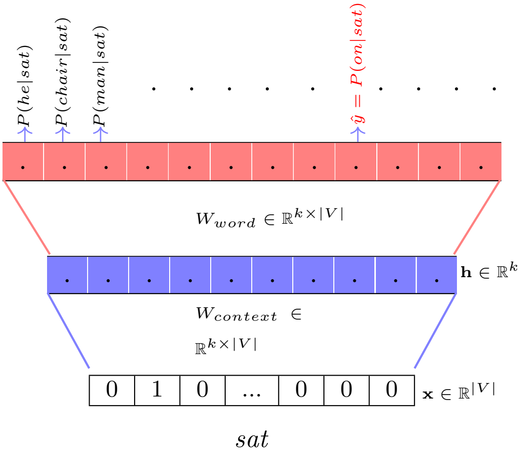
- How do we obtain $P(on|sat)$? For this multi-class classification problem what is an appropriate output function? (**softmax**)
- Therefore, $P(on|sat)$ is proportional to the dot product between j^{th} column of $\mathbf{W}_{context}$ and i^{th} column of $\mathbf{W}_{word}$
- $P(word = i|sat)$ thus depends on the i^{th} column of $\mathbf{W}_{word}$
- We thus treat the i-th column of $\mathbf{W}_{word}$ as the representation of word i
- Hope you see an analogy with SVD! (there we had a different way of learning $\mathbf{W}_{context}$ and $\mathbf{W}_{word}$ but we saw that the i^{th} column of $\mathbf{W}_{word}$ corresponded to the representation of the i^{th} word)
- Now that we understand the interpretation of $\mathbf{W}_{context}$ and $\mathbf{W}_{word}$, our aim now is to learn these parameters



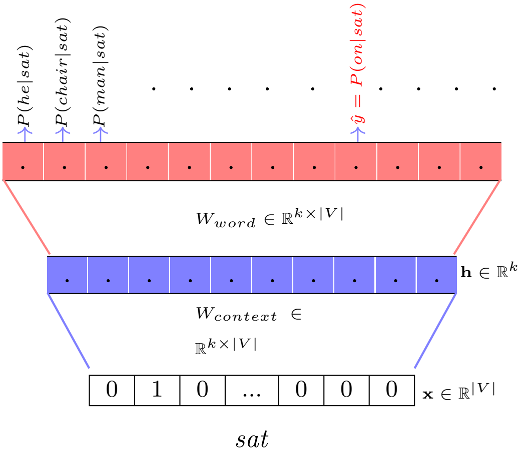
- We denote the context word (sat) by the index c and the correct output word (on) by the index w



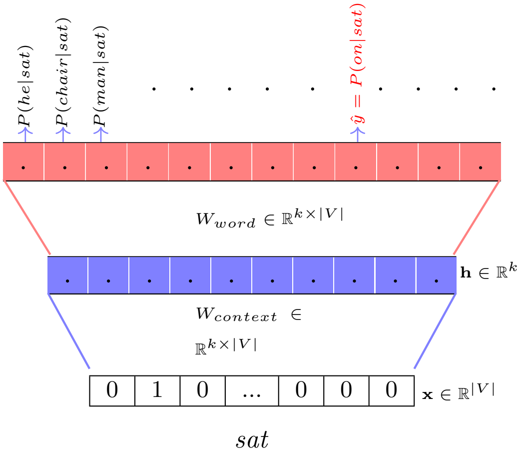
- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ?



- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**

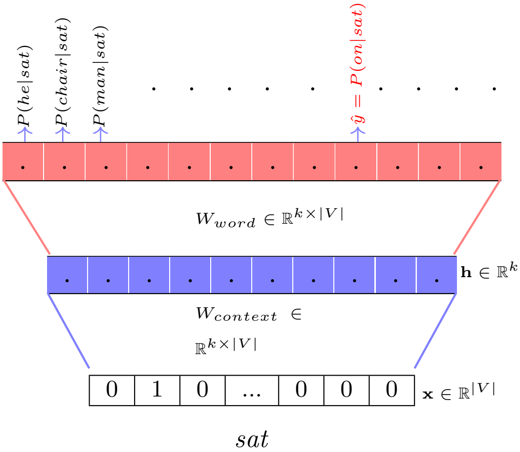


- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function?



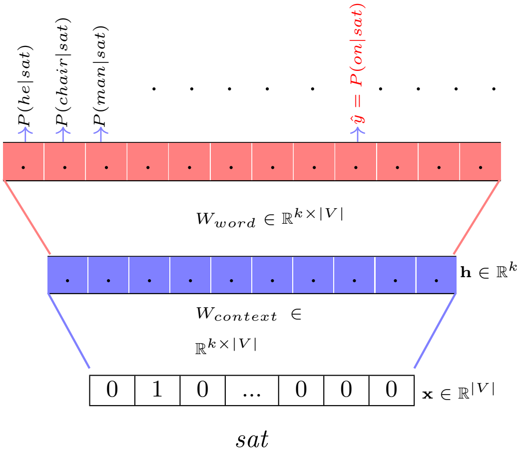
- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$



- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$
$$h = W_{context} \cdot x_c = u_c$$

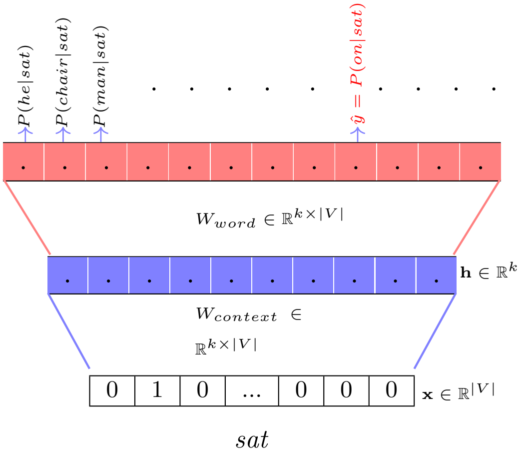


- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$

$$h = W_{context} \cdot x_c = u_c$$

$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$



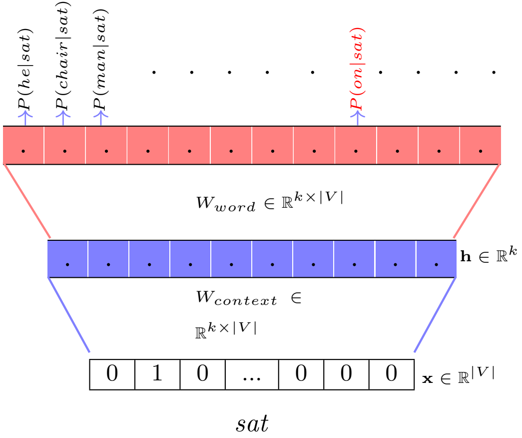
- We denote the context word (sat) by the index c and the correct output word (on) by the index w
- For this multiclass classification problem what is an appropriate output function ($\hat{y} = f(x)$) ? **softmax**
- What is an appropriate loss function? **cross entropy**

$$\mathscr{L}(\theta) = -\log \hat{y}_w = -\log P(w|c)$$

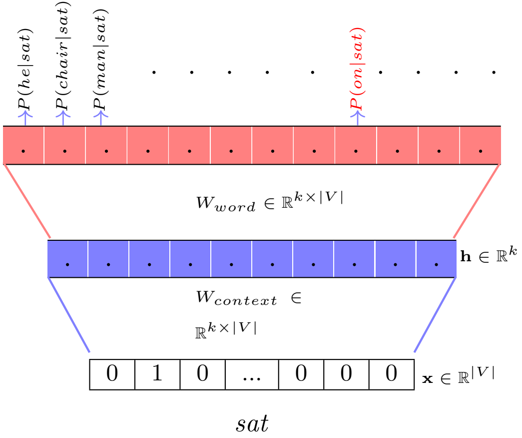
$$h = W_{context} \cdot x_c = u_c$$

$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

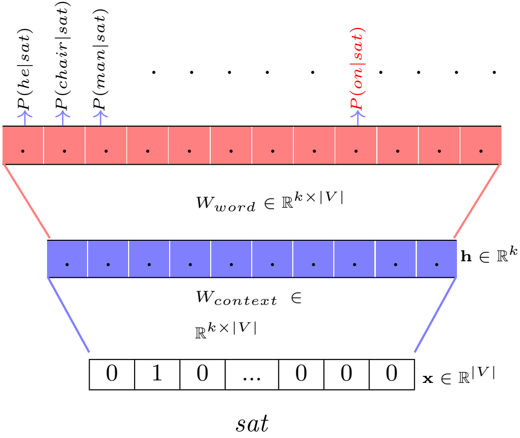
u_c is the column of $W_{context}$ corresponding to context c and v_w is the column of W_{word} corresponding to context w



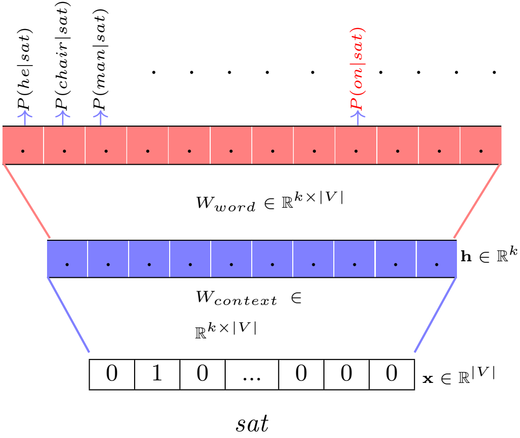
- How do we train this simple feed forward neural network?



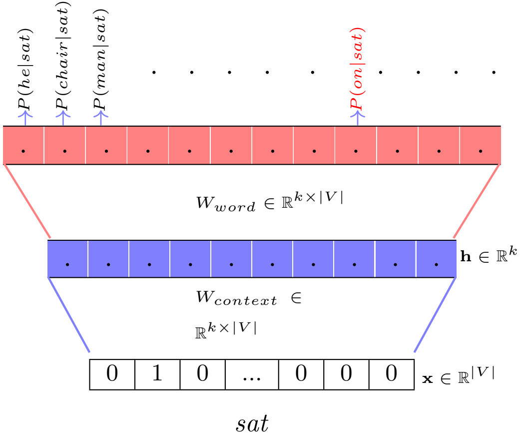
- How do we train this simple feed forward neural network? backpropagation



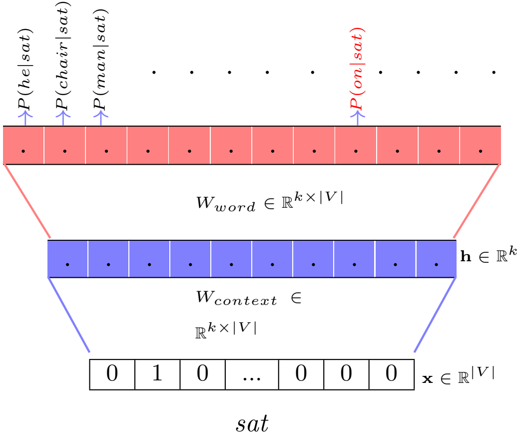
- How do we train this simple feed forward neural network? backpropagation
- Let us consider one input-output pair (c, w) and see the update rule for v_w



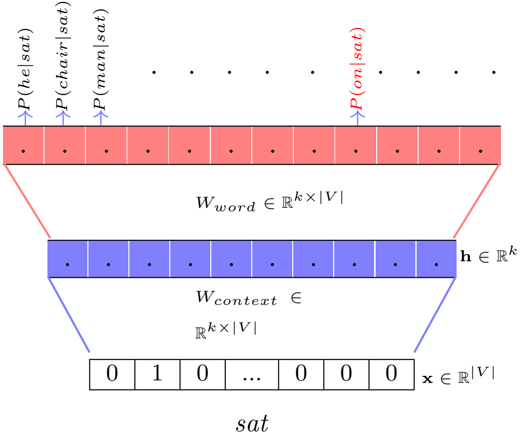
$$\mathscr{L}(\theta) = -\log \hat{y}_w$$



$$\mathscr{L}(\theta) = -\log \hat{y}_w$$
$$= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$



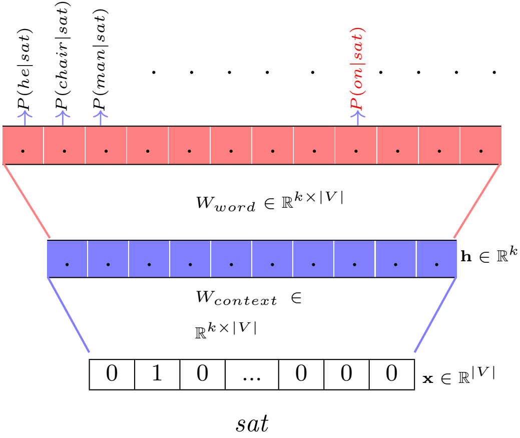
$$\mathscr{L}(\theta) = -\log \hat{y}_w$$
$$= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$
$$= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))$$



$$\nabla_{v_w} = -\frac{\partial}{\partial v_w}\mathscr{L}(\theta)$$

$$\begin{aligned}\mathscr{L}(\theta) &= -\log \hat{y}_w \\ &= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \\ &= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))\end{aligned}$$

$$\nabla_{v_w} = -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c)$$



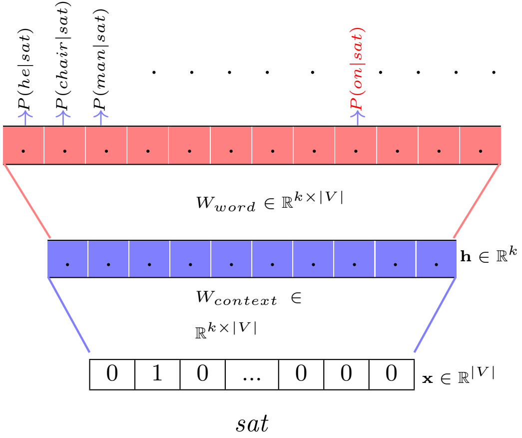
$$\mathscr{L}(\theta) = -\log \hat{y}_w$$

$$= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

$$= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))$$

$$\nabla_{v_w} = -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c)$$

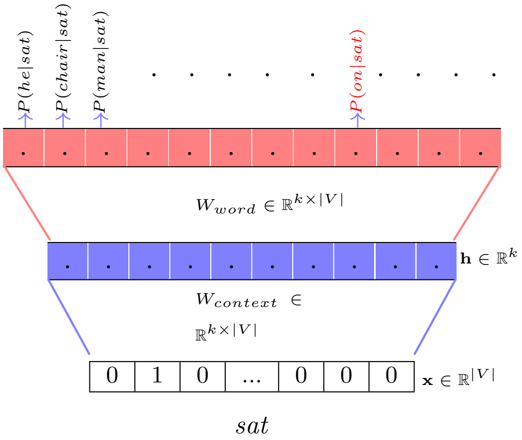
$$= -u_c(1 - \hat{y}_w)$$



$$
\begin{aligned}
\mathscr{L}(\theta) &= -\log \hat{y}_w \\
&= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \\
&= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))
\end{aligned}
$$

$$
\begin{aligned}
\nabla_{v_w} &= -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c) \\
&= -u_c(1 - \hat{y}_w)
\end{aligned}
$$

And the update rule would be



$$\mathscr{L}(\theta) = -\log \hat{y}_w$$

$$= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

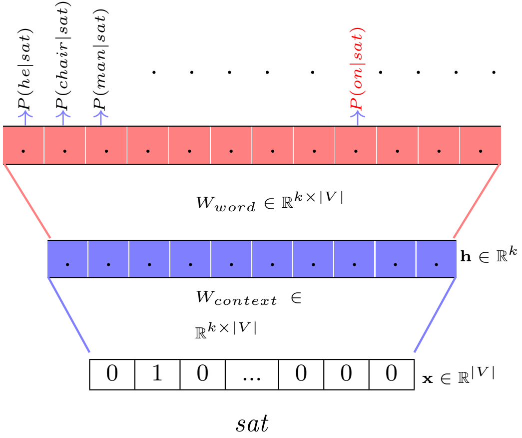
$$= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))$$

$$\nabla_{v_w} = -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c)$$

$$= -u_c(1 - \hat{y}_w)$$

And the update rule would be

$$v_w = v_w - \eta \nabla_{v_w}$$

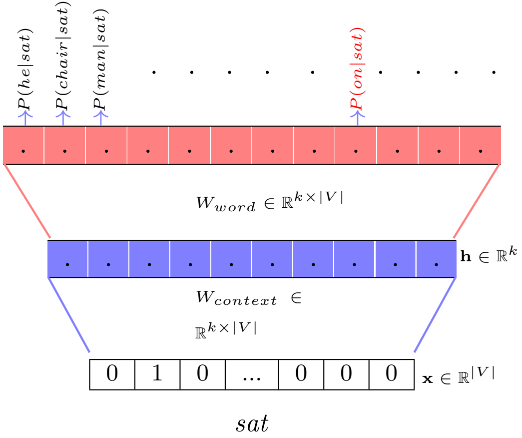


$$\begin{aligned}
\mathscr{L}(\theta) &= -\log \hat{y}_w \\
&= -\log \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \\
&= -(u_c \cdot v_w - \log \sum_{w' \in V} exp(u_c \cdot v_{w'}))
\end{aligned}$$

$$\begin{aligned}
\nabla_{v_w} &= -(u_c - \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})} \cdot u_c) \\
&= -u_c(1 - \hat{y}_w)
\end{aligned}$$

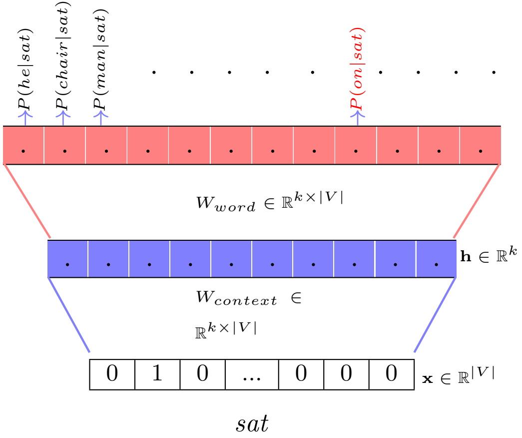
And the update rule would be

$$\begin{aligned}
v_w &= v_w - \eta \nabla_{v_w} \\
&= v_w + \eta u_c(1 - \hat{y}_w)
\end{aligned}$$



- This update rule has a nice interpretation

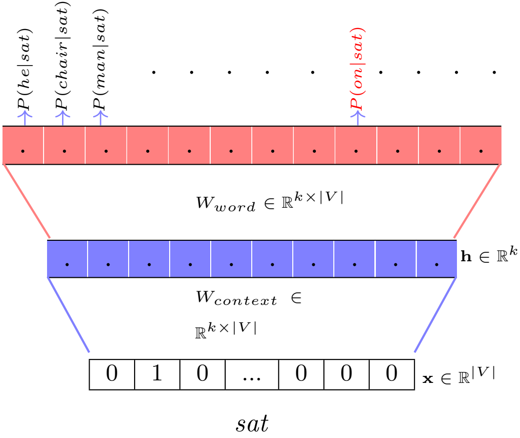
$$v_w = v_w + \eta u_c(1 - \hat{y}_w)$$



- This update rule has a nice interpretation

$$v_w = v_w + \eta u_c(1 - \hat{y}_w)$$

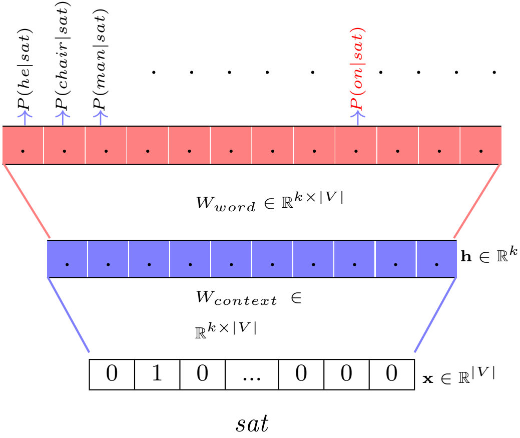
- If $\hat{y}_w \to 1$ then we are already predicting the right word and v_w will not be updated



- This update rule has a nice interpretation

$$v_w = v_w + \eta u_c(1 - \hat{y}_w)$$

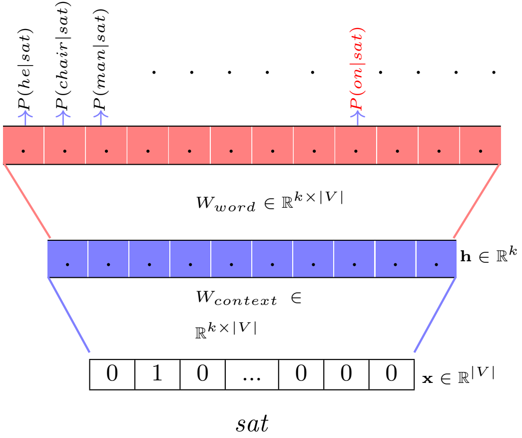
- If $\hat{y}_w \to 1$ then we are already predicting the right word and v_w will not be updated
- If $\hat{y}_w \to 0$ then v_w gets updated by adding a fraction of u_c to it



- This update rule has a nice interpretation

$$v_w = v_w + \eta u_c(1 - \hat{y}_w)$$

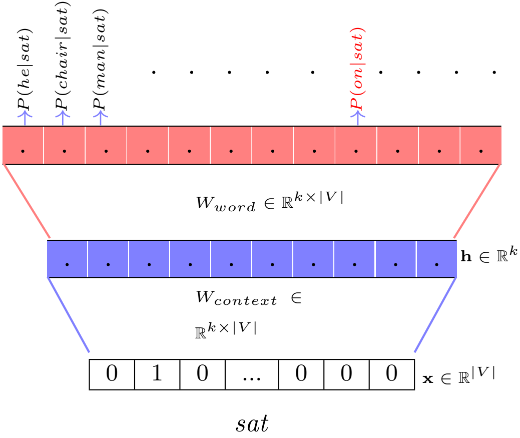
- If $\hat{y}_w \to 1$ then we are already predicting the right word and v_w will not be updated
- If $\hat{y}_w \to 0$ then v_w gets updated by adding a fraction of u_c to it
- This increases the cosine similarity between v_w and u_c (How? Refer to slide 38 of Lecture 2)



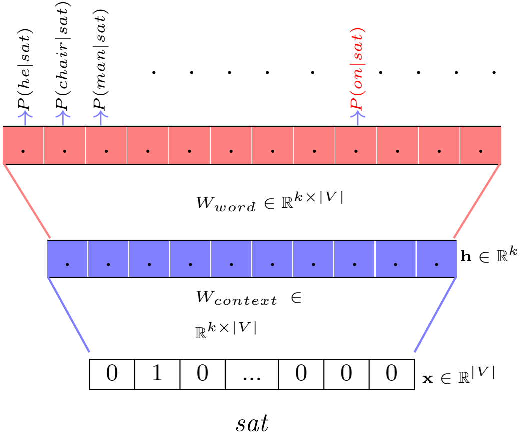
- This update rule has a nice interpretation

$$v_w = v_w + \eta u_c(1 - \hat{y}_w)$$

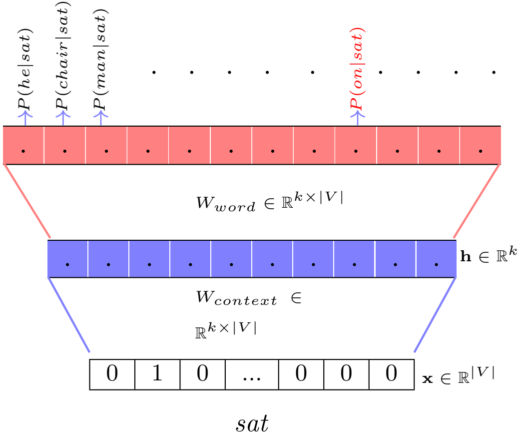
- If $\hat{y}_w \to 1$ then we are already predicting the right word and v_w will not be updated
- If $\hat{y}_w \to 0$ then v_w gets updated by adding a fraction of u_c to it
- This increases the cosine similarity between v_w and u_c (How? Refer to slide 38 of Lecture 2)
- The training objective ensures that the cosine similarity between word (v_w) and context word (u_c) is maximized



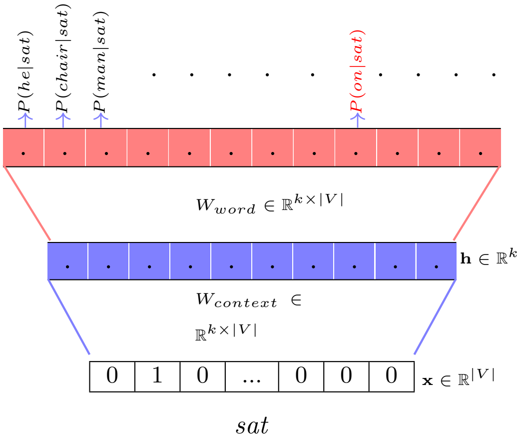
- What happens to the representations of two words w and w' which tend to appear in similar context (c)



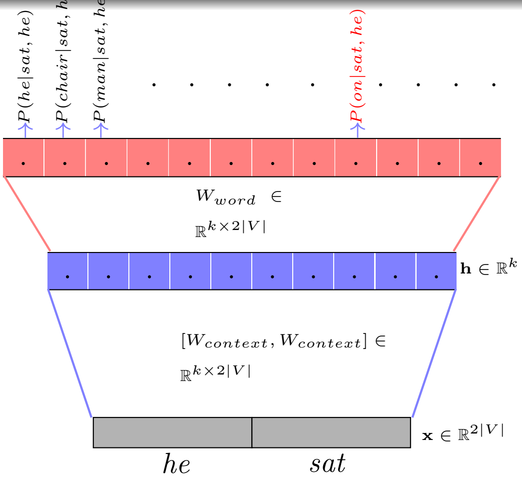
- What happens to the representations of two words w and w' which tend to appear in similar context (c)
- The training ensures that both v_w and v'_w have a high cosine similarity with u_c and hence transitively (intuitively) ensures that v_w and v'_w have a high cosine similarity with each other



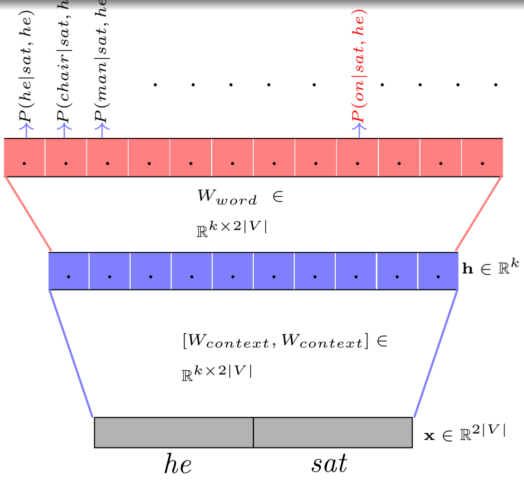
- What happens to the representations of two words w and w' which tend to appear in similar context (c)
- The training ensures that both v_w and v'_w have a high cosine similarity with u_c and hence transitively (intuitively) ensures that v_w and v'_w have a high cosine similarity with each other
- This is only an intuition (reasonable)



- What happens to the representations of two words w and w' which tend to appear in similar context (c)
- The training ensures that both v_w and v'_w have a high cosine similarity with u_c and hence transitively (intuitively) ensures that v_w and v'_w have a high cosine similarity with each other
- This is only an intuition (reasonable)
- Haven't come across a formal proof for this!

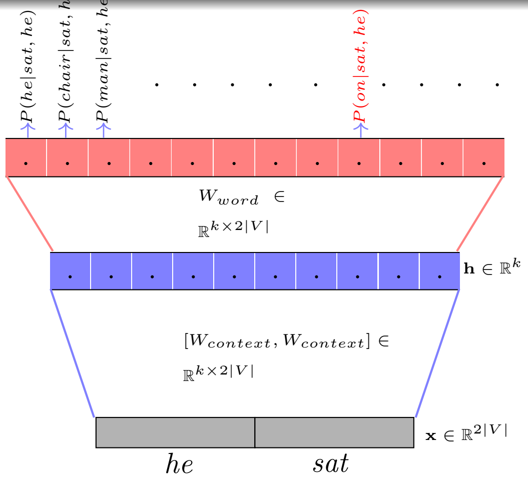


- In practice, instead of window size of 1 it is common to use a window size of d



- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

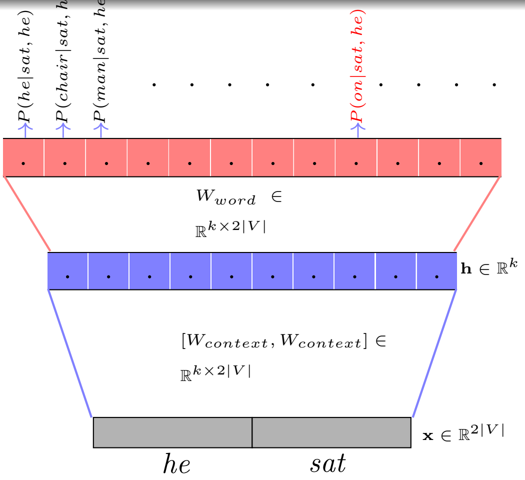
$$h = \sum_{i=1}^{d-1} u_{c_i}$$



- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

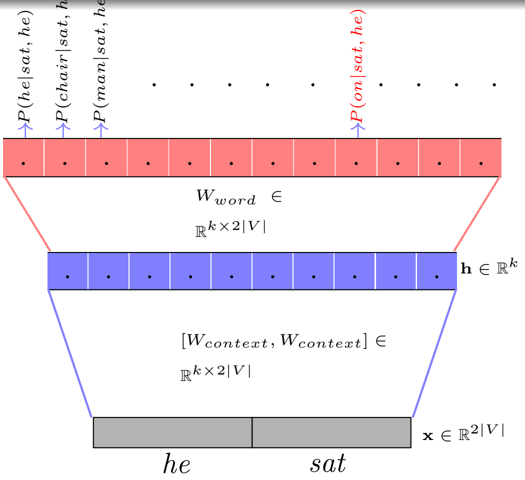


- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & & & \\ 3 & -1 & -2 & & & \\ -2 & 1.7 & 3 & & & \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \} sat \\ \\ \\ \\ \} he \end{matrix}$$

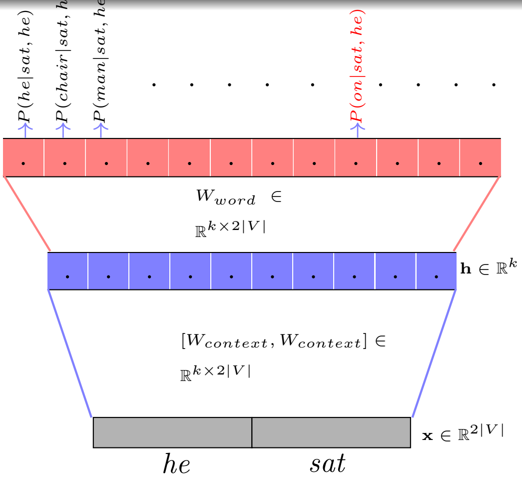


- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & -1 & 0.5 & 2 \\ 3 & -1 & -2 & 3 & -1 & -2 \\ -2 & 1.7 & 3 & -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \} \, sat \\ \\ \\ \\ \} \, he \end{matrix}$$



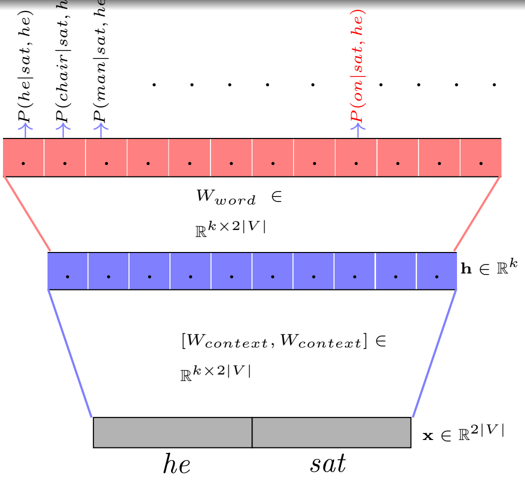
- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & -1 & 0.5 & 2 \\ 3 & -1 & -2 & 3 & -1 & -2 \\ -2 & 1.7 & 3 & -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \} sat \\ \\ \\ \\ \} he \end{matrix}$$

$$= \begin{bmatrix} 2.5 \\ -3 \\ 4.7 \end{bmatrix}$$



- In practice, instead of window size of 1 it is common to use a window size of d
- So now,

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- $[W_{context}, W_{context}]$ just means that we are stacking 2 copies of $W_{context}$ matrix

$$\begin{bmatrix} -1 & 0.5 & 2 & -1 & 0.5 & 2 \\ 3 & -1 & -2 & 3 & -1 & -2 \\ -2 & 1.7 & 3 & -2 & 1.7 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} \\ \} \, sat \\ \\ \\ \\ \} \, he \end{matrix}$$

$$= \begin{bmatrix} 2.5 \\ -3 \\ 4.7 \end{bmatrix}$$

- The resultant product would simply be the sum of the columns corresponding to 'sat' and 'he'

- Of course in practice we will not do this expensive matrix multiplication

- Of course in practice we will not do this expensive matrix multiplication
- If 'he' is i^{th} word in the vocabulary and *sat* is the j^{th} word then we will simply access columns $\mathbf{W}[i:]$ and $\mathbf{W}[j:]$ and add them

- Now what happens during backpropagation

- Now what happens during backpropagation
- Recall that

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- Now what happens during backpropagation
- Recall that

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- and

$$P(on|sat, he) = \frac{e^{(w_{word}h)[k]}}{\sum_j e^{(w_{word}h)[j]}}$$

- Now what happens during backpropagation
- Recall that

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- and

$$P(on|sat, he) = \frac{e^{(w_{word}h)[k]}}{\sum_j e^{(w_{word}h)[j]}}$$

- where 'k' is the index of the word 'on'

- Now what happens during backpropagation
- Recall that

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- and

$$P(on|sat, he) = \frac{e^{(w_{word}h)[k]}}{\sum_j e^{(w_{word}h)[j]}}$$

- where 'k' is the index of the word 'on'
- The loss function depends on $\{W_{word}, u_{c_1}, u_{c_2}, \ldots, u_{c_{d-1}}\}$ and all these parameters will get updated during backpropogation

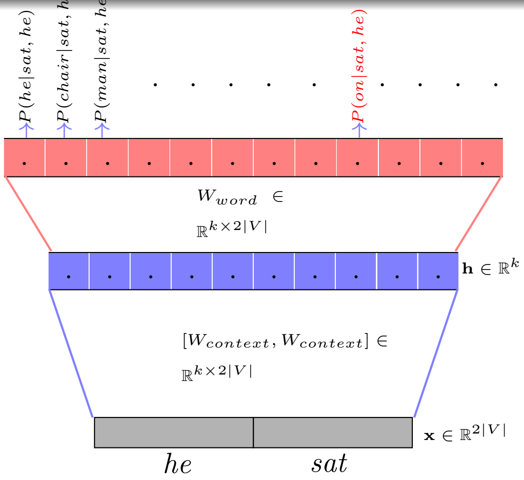
- Now what happens during backpropagation
- Recall that

$$h = \sum_{i=1}^{d-1} u_{c_i}$$

- and

$$P(on|sat, he) = \frac{e^{(w_{word}h)[k]}}{\sum_j e^{(w_{word}h)[j]}}$$

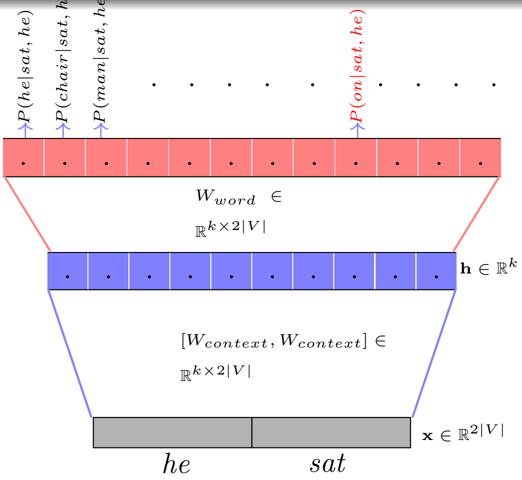
- where 'k' is the index of the word 'on'
- The loss function depends on $\{W_{word}, u_{c_1}, u_{c_2}, \ldots, u_{c_{d-1}}\}$ and all these parameters will get updated during backpropogation
- Try deriving the update rule for v_w now and see how it differs from the one we derived before



Some problems:

- Notice that the softmax function at the output is computationally very expensive

$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

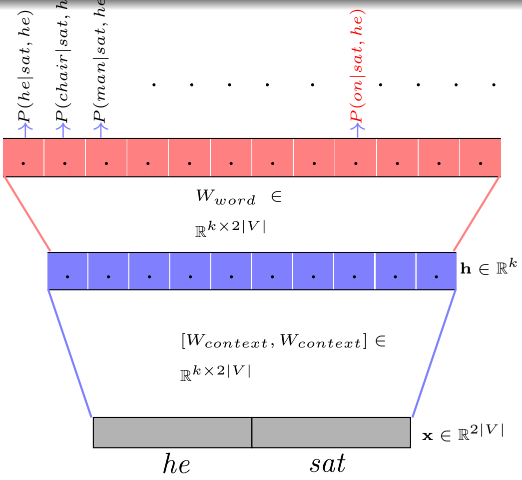


Some problems:

- Notice that the softmax function at the output is computationally very expensive

$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

- The denominator requires a summation over all words in the vocabulary



Some problems:

- Notice that the softmax function at the output is computationally very expensive

$$\hat{y}_w = \frac{exp(u_c \cdot v_w)}{\sum_{w' \in V} exp(u_c \cdot v_{w'})}$$

- The denominator requires a summation over all words in the vocabulary
- We will revisit this issue soon