Word2vec

박준형
데이터인텔리전스 연구실

irish07@korea.ac.kr

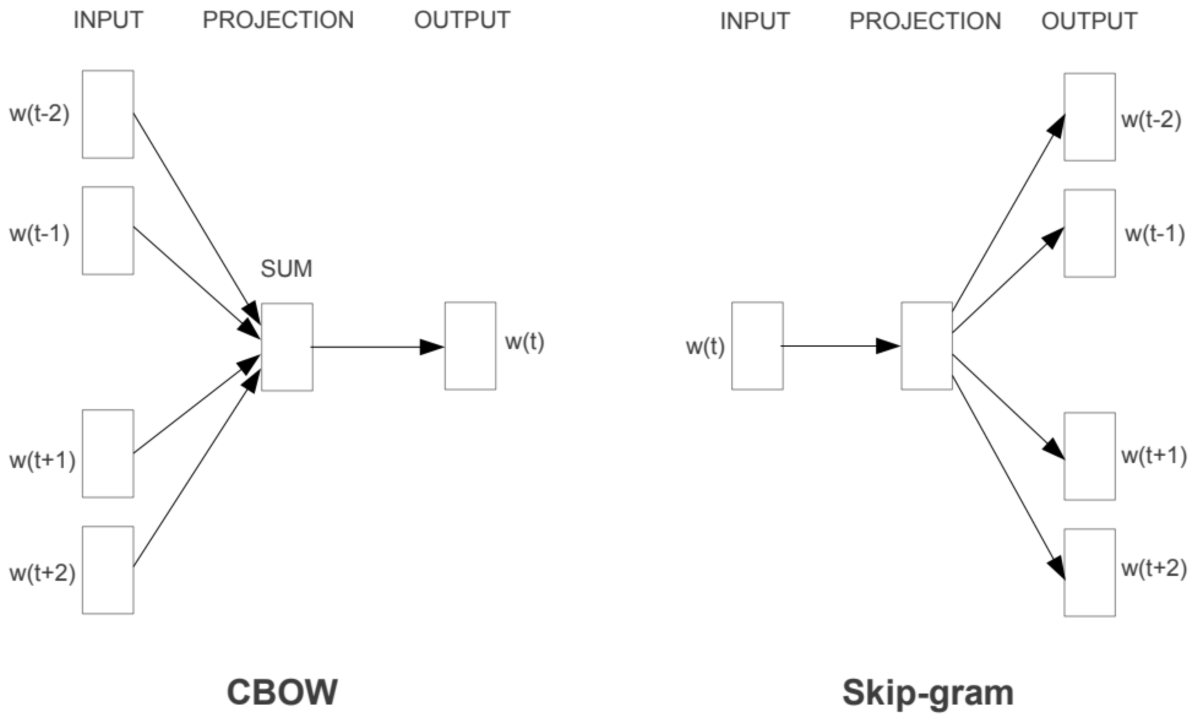
Class Lab - Schedule & Assignment

1. Skip-gram / CBOW (~5/20)
   (Basic) Softmax

2. Hierarchical Softmax / Negative sampling (~6/7)
   Subsampling

Class Lab - Schedule & Assignment

- T. Mikolov, K. Chen, G. Corrado, J. Dean, “Efficient Estimation of Word Representations in Vector Space”, ICLR 2013



Class Lab - Schedule & Assignment

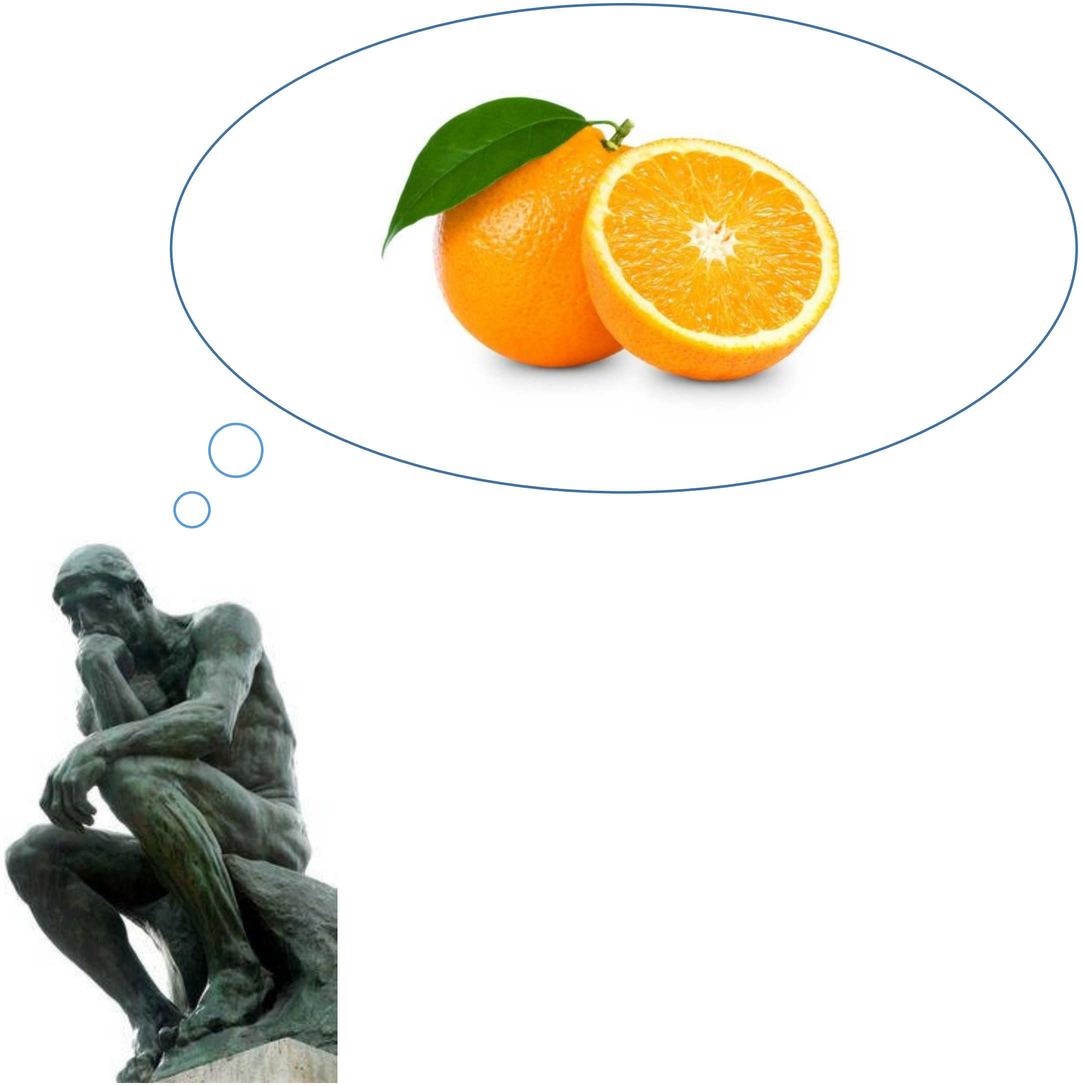
- T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, “Distributed Representations of Words and Phrases and their Compositionality”, NIPS 2013

Method	Time [min]	Syntactic [%]	Semantic [%]	Total accuracy [%]
NEG-5	38	63	54	59
NEG-15	97	63	58	**61**
HS-Huffman	41	53	40	47
NCE-5	38	60	45	53
The following results use 10^{-5} subsampling				
NEG-5	14	61	58	60
NEG-15	36	61	61	**61**
HS-Huffman	21	52	59	55

How to represent words

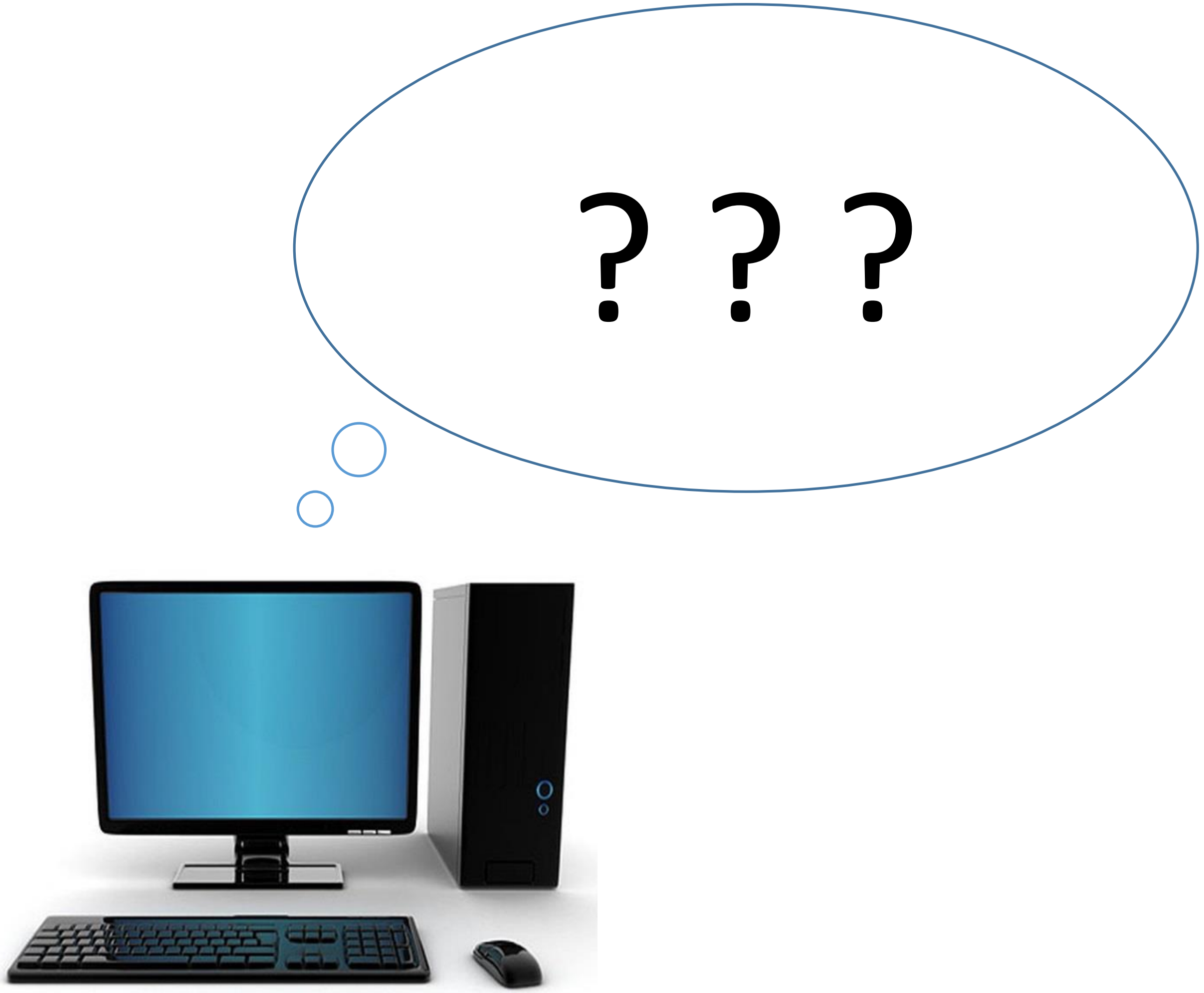
Word Representation

What is “Orange”? ➡

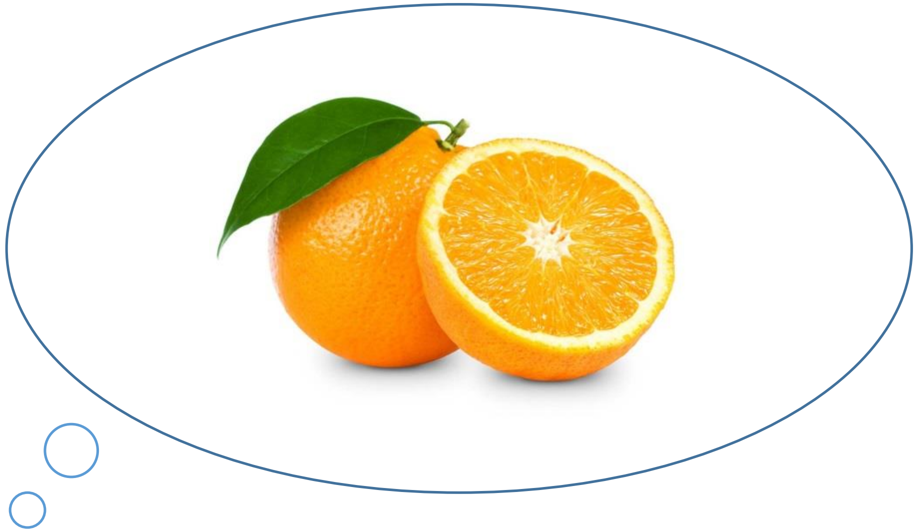


Word Representation

What is “Orange”?



Word Representation



Orange

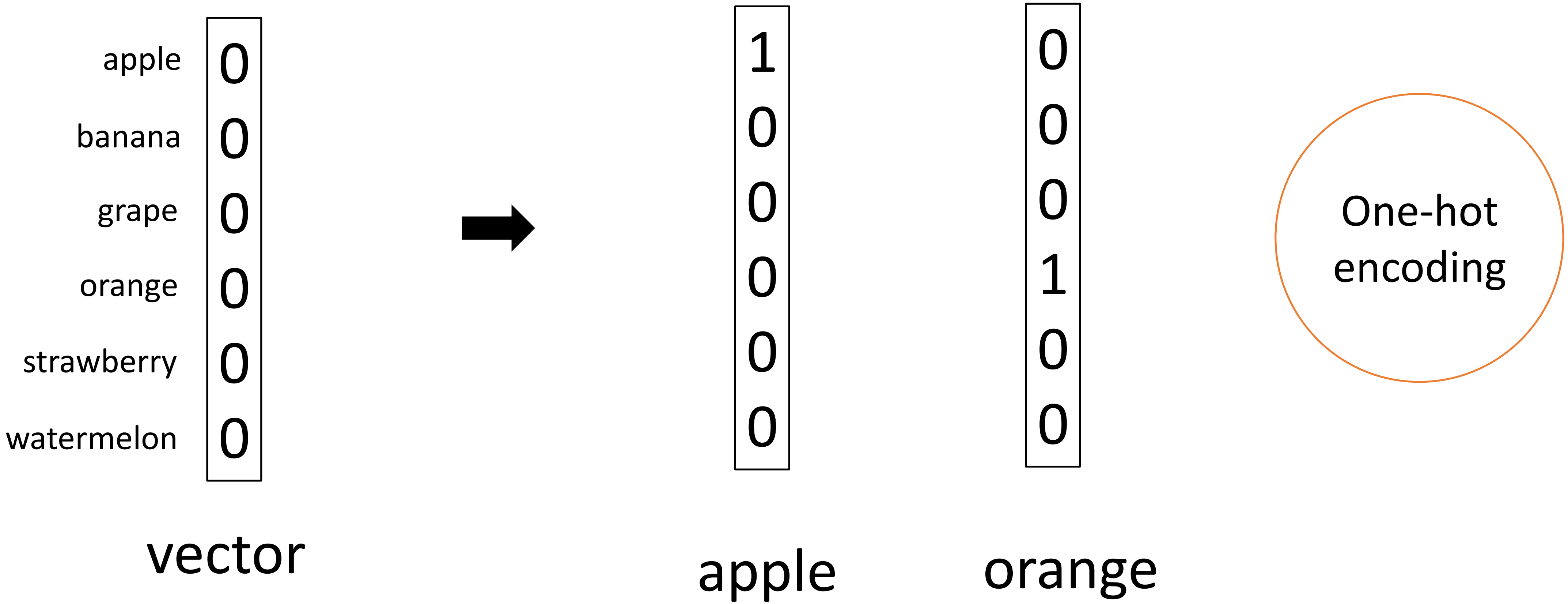
↓

Representation

→

Word Representation

Atomic Word Representation



Word Representation

Atomic Word Representation

All vectors are independent

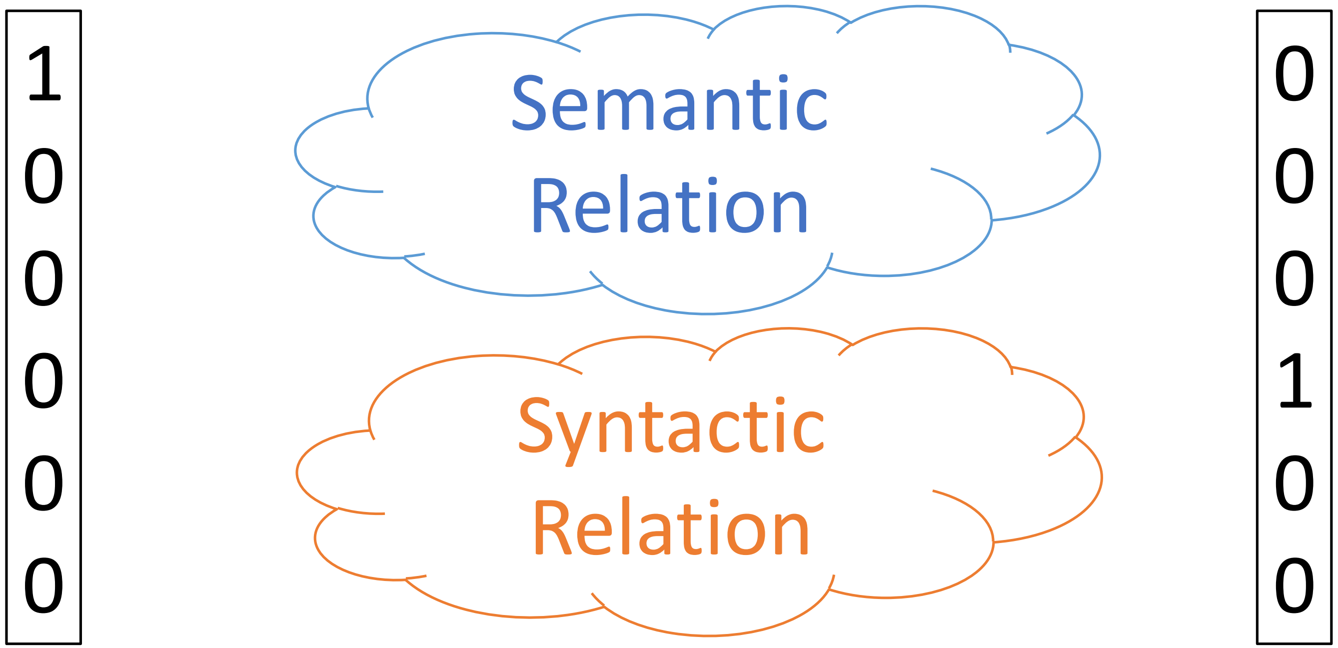
$$\begin{bmatrix}1\\0\\0\\0\\0\\0\end{bmatrix} \qquad\qquad \begin{bmatrix}0\\0\\0\\1\\0\\0\end{bmatrix}$$

apple orange

Word Representation

Atomic Word Representation

But words are dependent

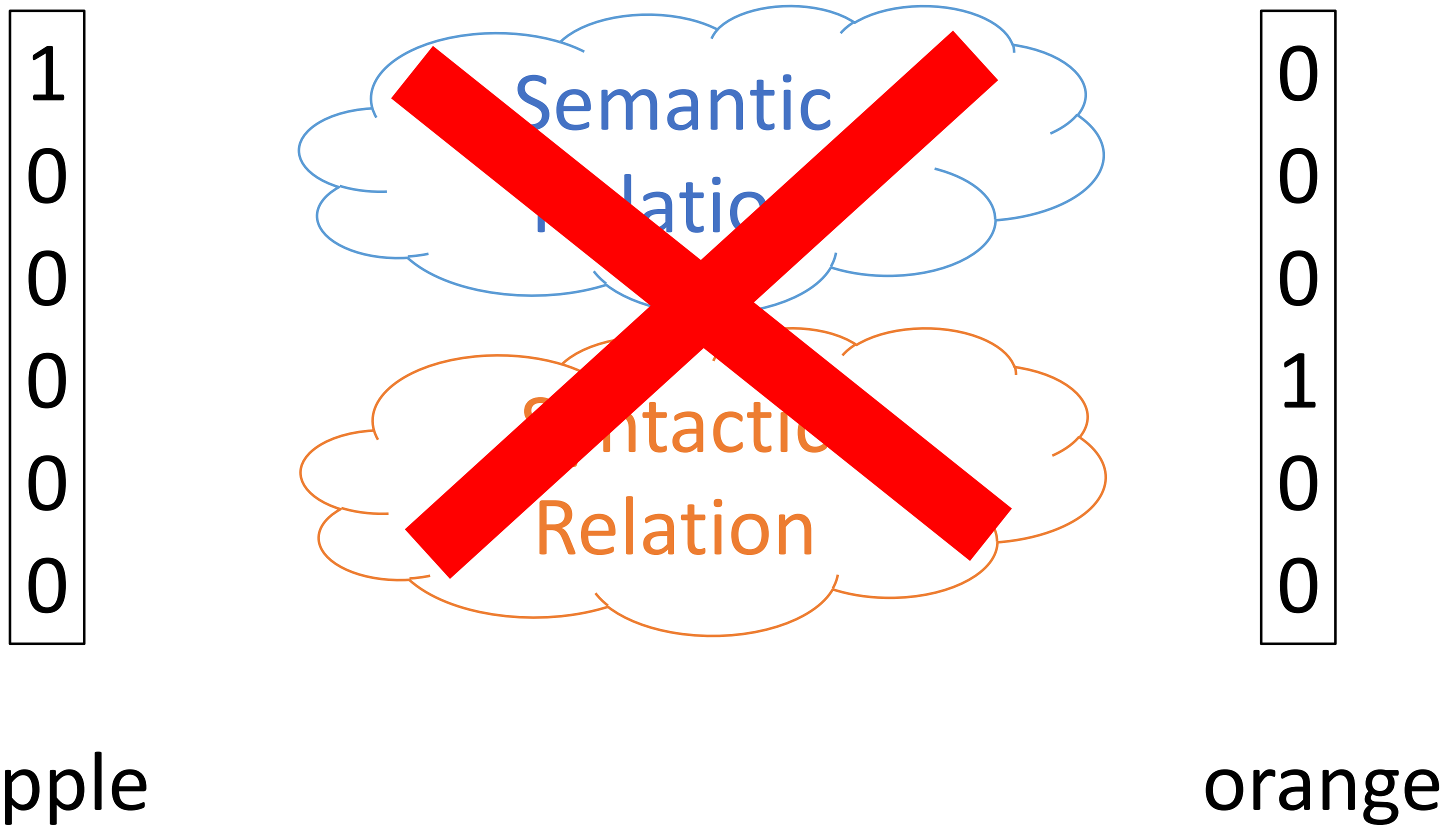


apple

orange

Word Representation

Atomic Word Representation



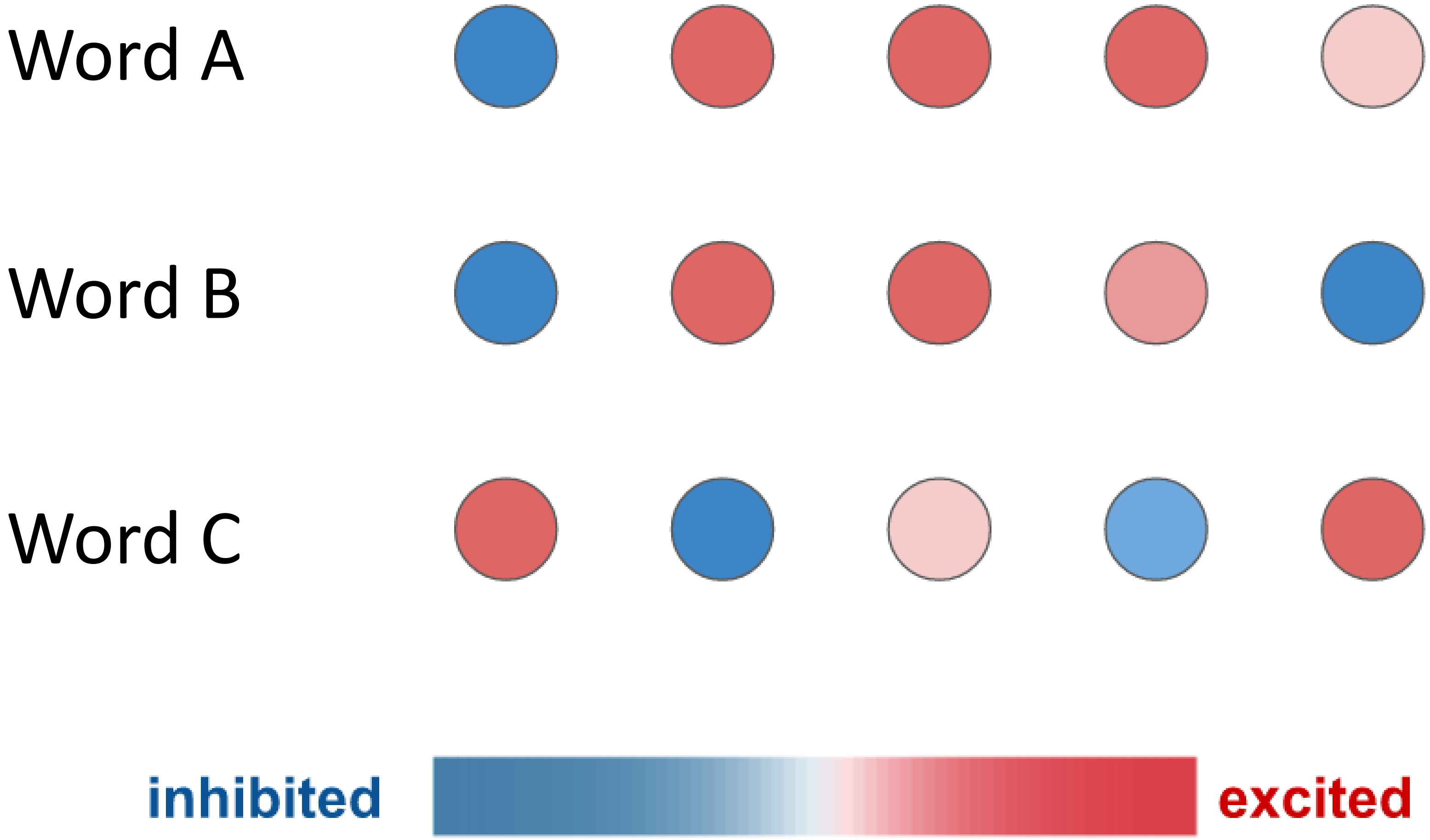
Word Representation

Distributed Representation

Word ➡ Continuous Feature Space

Word Representation

Distributed Representation



Word Representation

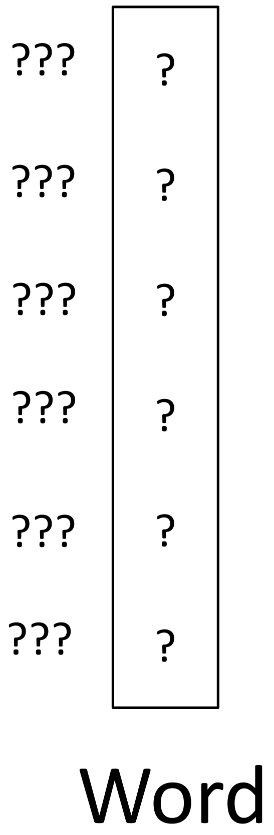
Distributed Representation

Word Representation

Distributed Representation

Word Representation

Distributed Representation



How to set features and values of them

Word Representation

Distributed Representation

???	?
???	?
???	?
???	?
???	?
???	?

Word

How to set features and values of them

Manually?

Word Representation

Distributed Representation

???	?
???	?
???	?
???	?
???	?
???	?

Word

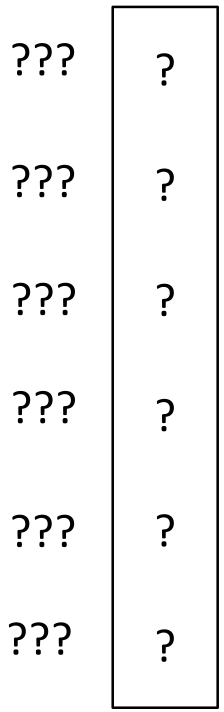
How to set features and values of them

English words

- More than 1 million
- 1 new word every 98 minutes

Word Representation

Distributed Representation



Word

How to set features and values of them

Manually? Impossible

Word Representation

Distributed Representation

???	?
???	?
???	?
???	?
???	?
???	?

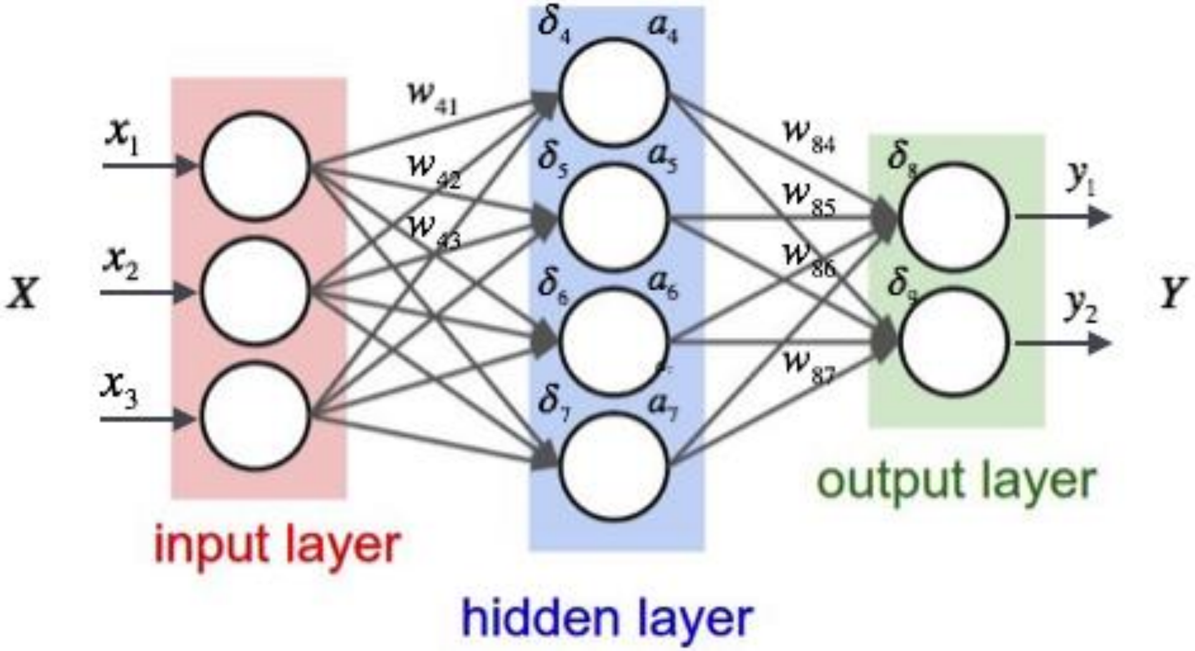
Word

How to set features and values of them

With Neural Networks

Neural Network

Automatically Detect Features



Neural Network

Ex) Obesity with height and weight

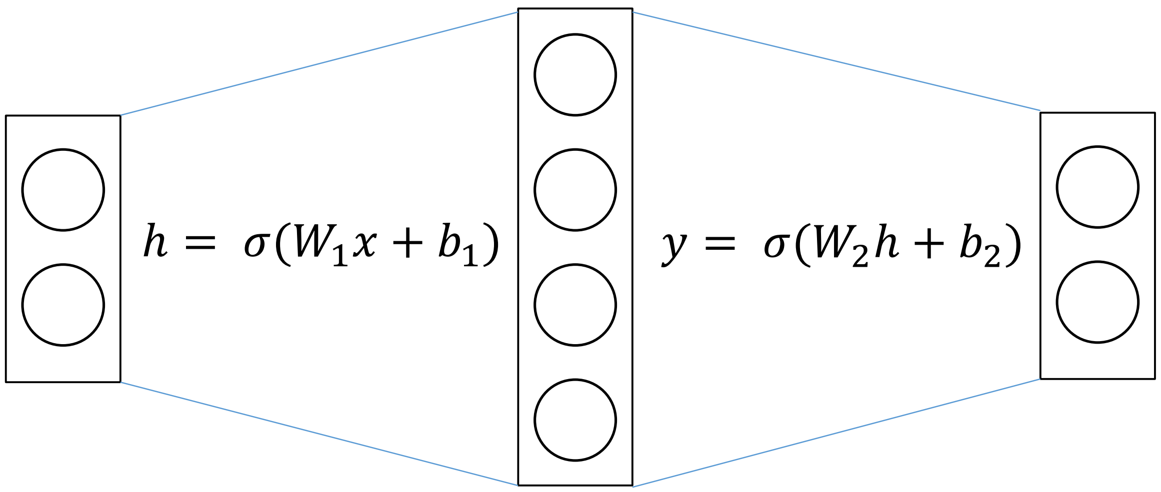
$$BMI = weight \ / \ height^2$$

BMI > 30 : Obesity
BMI <= 30 : Normal

Height(m)	Weight(kg)	Obesity
1.81	70	False
1.63	68	False
1.75	95	True
1.55	46	False
1.78	103	True

Neural Network

Ex) Obesity with height and weight

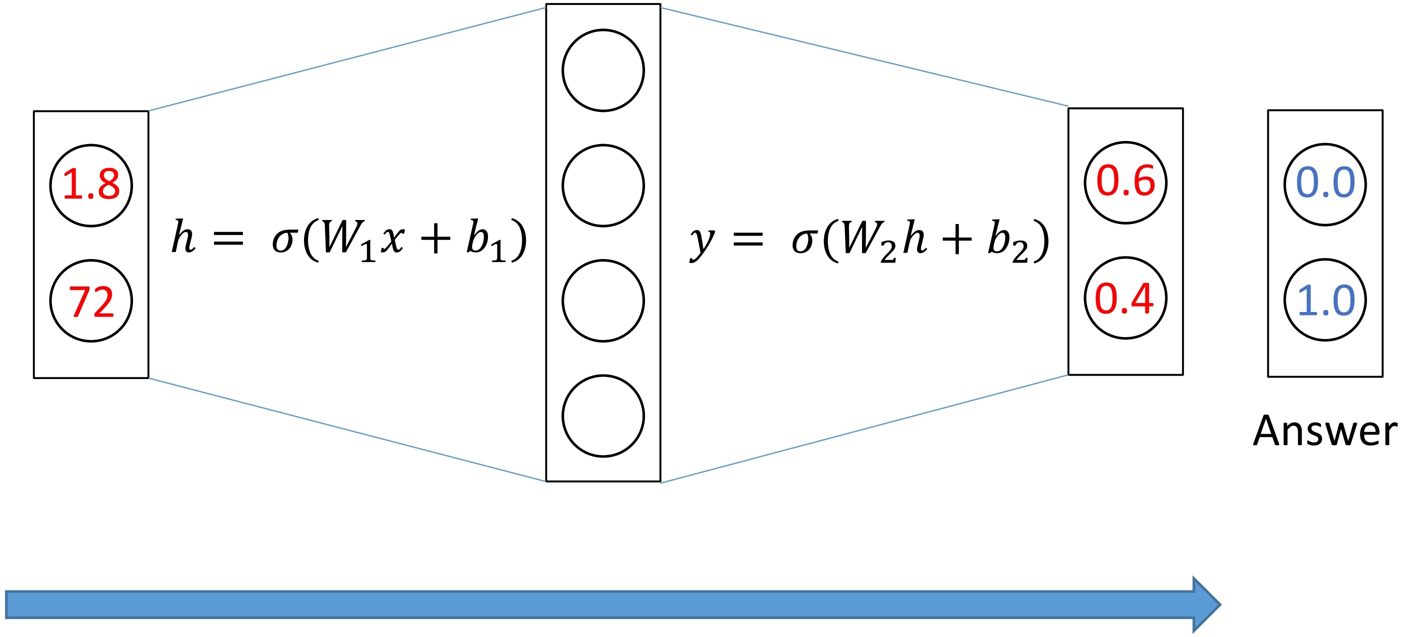


Input : Height, Weight
Output : Probabilities of True or False

Neural network
with randomly initialized parameters

Neural Network

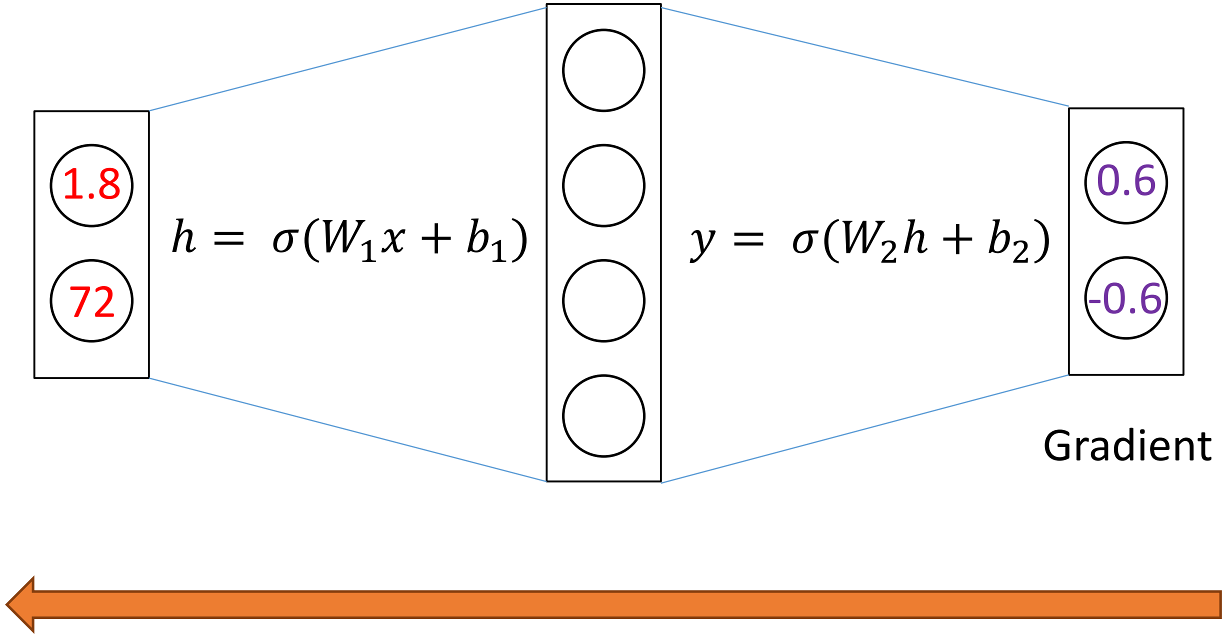
Ex) Obesity with height and weight



The first output will be very different

Neural Network

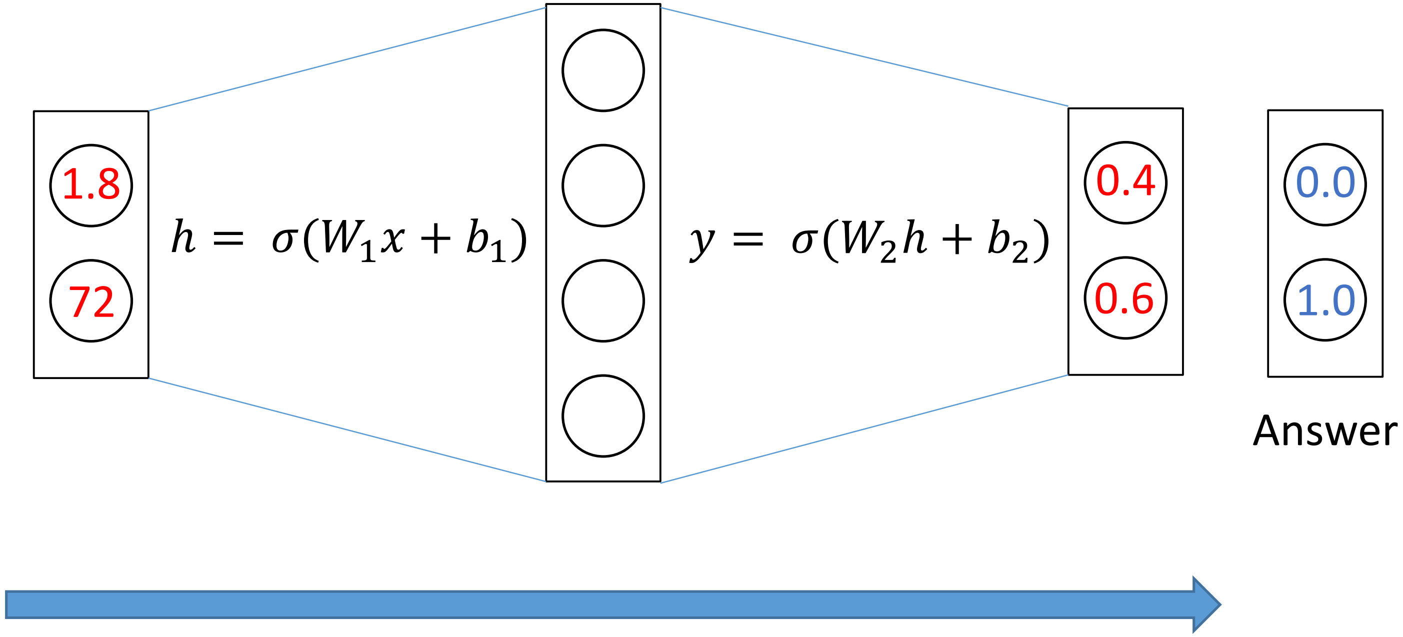
Ex) Obesity with height and weight



Update parameters with Backpropagation and Gradient descent

Neural Network

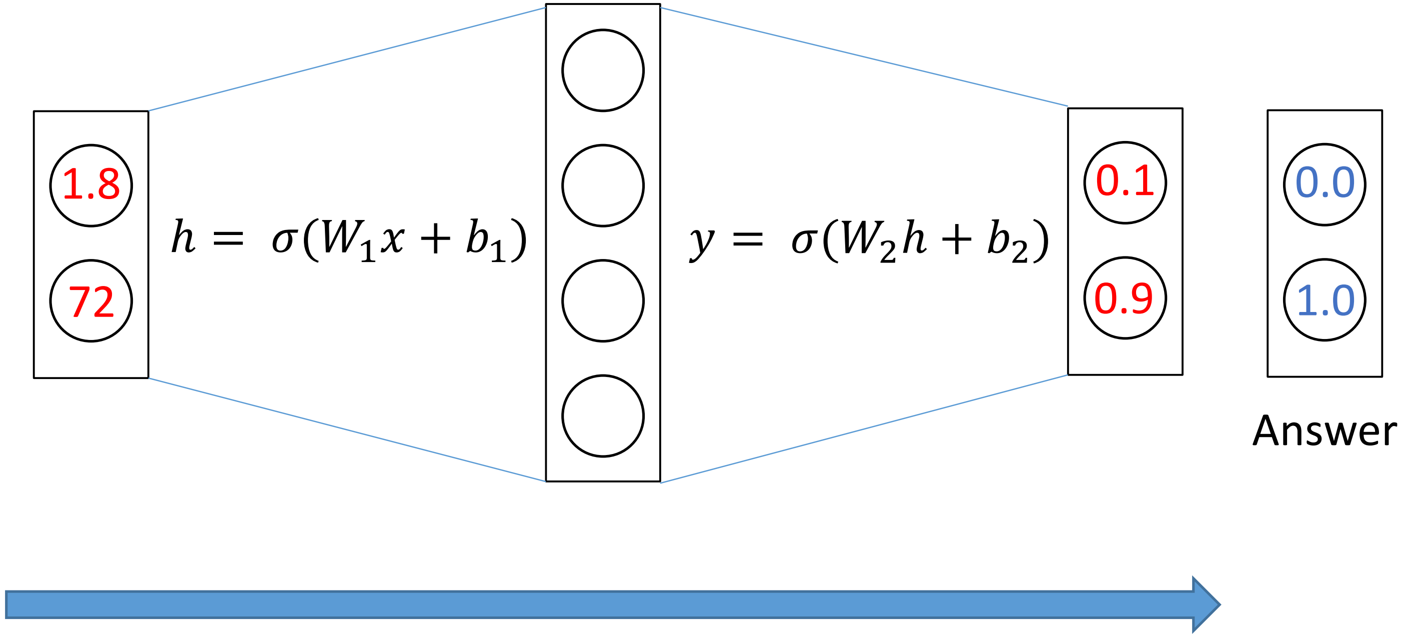
Ex) Obesity with height and weight



The second output will be closer to the answer

Neural Network

Ex) Obesity with height and weight

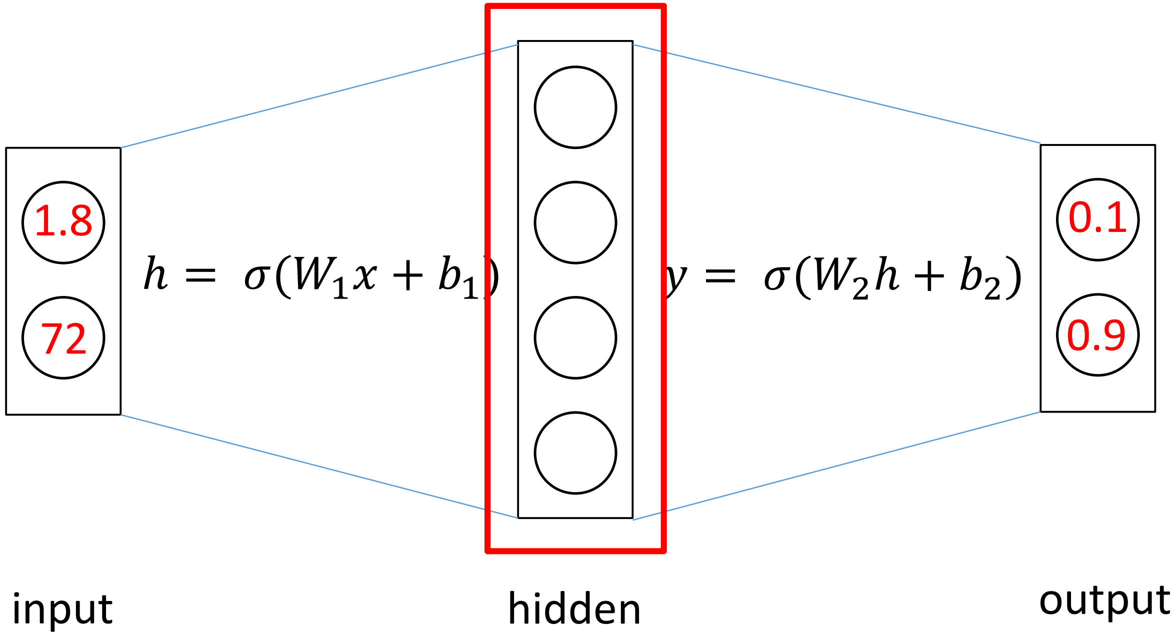


After repeat of training,
the neural network will
approximate the obesity function

Neural Network

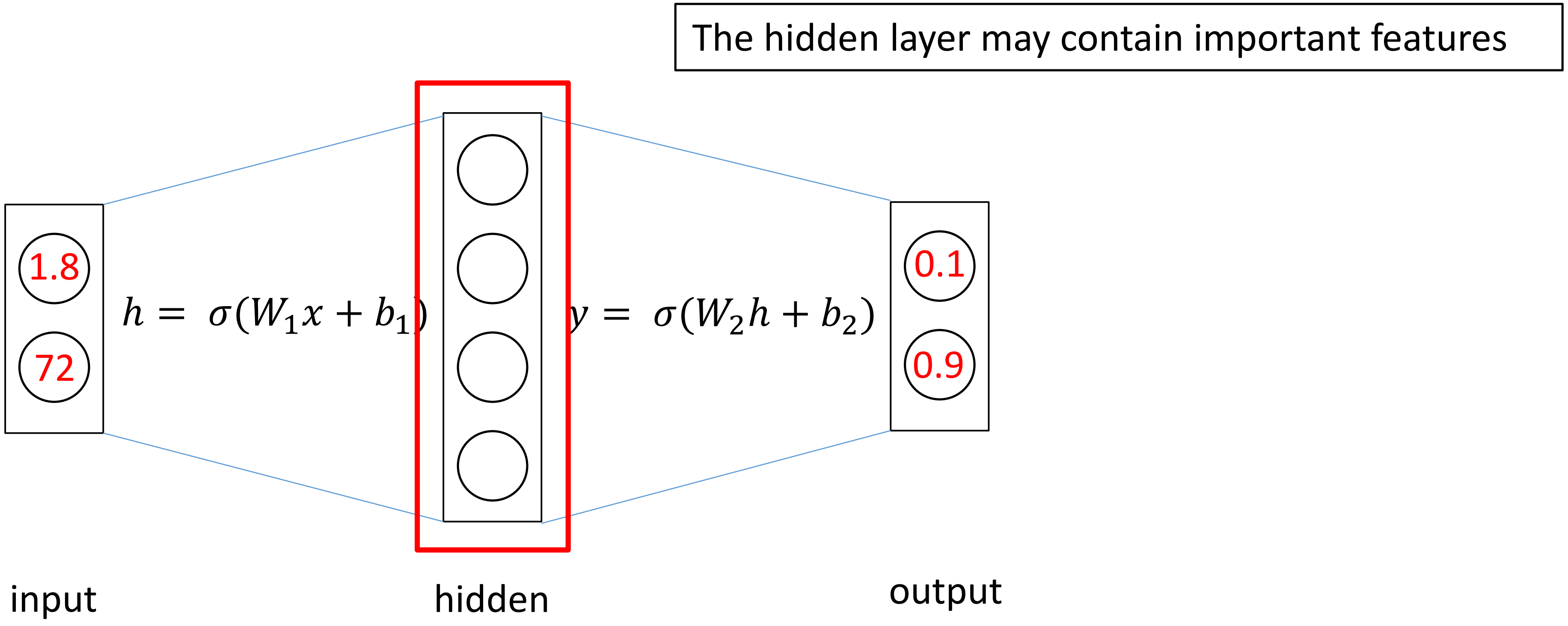
Ex) Obesity with height and weight

Output is calculated based on **the hidden layer**

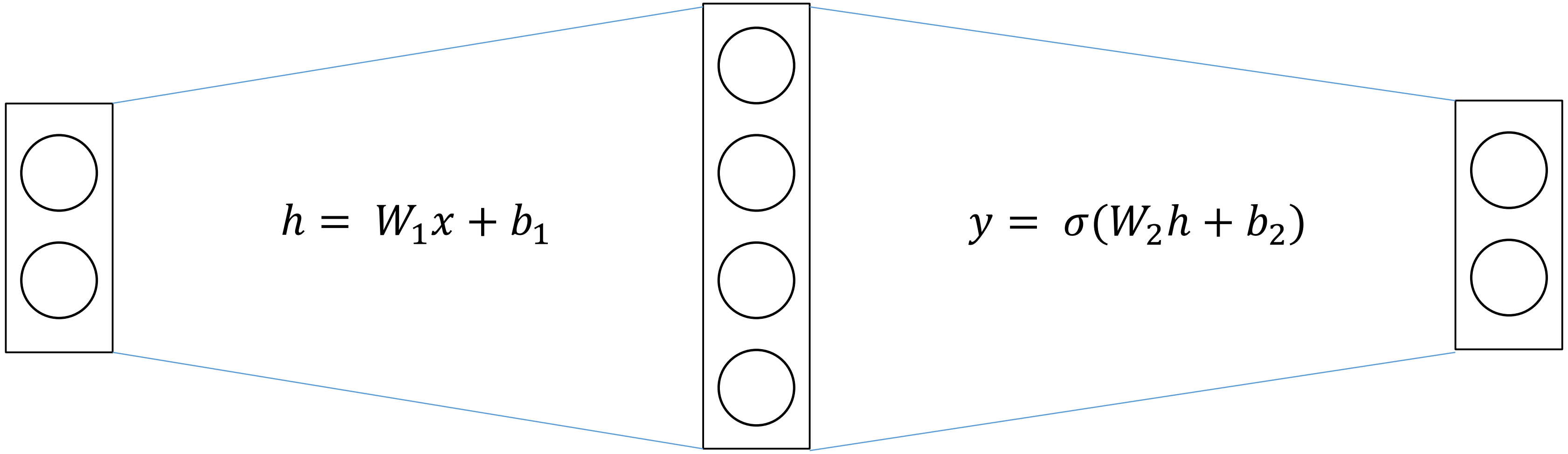


Neural Network

Ex) Obesity with height and weight



Word2Vec



Input → Word Representation → Output

Automatically determined

Word
(one-hot vector)

???

Word2Vec

I eat an apple every day (O)

I eat an orange every day (O)

I eat a car every day (X)

Word2Vec

I eat an apple every day (O)

I eat an orange every day (O)

I eat a car every day (X)

Word2Vec

I eat an apple every day (O) → Co-occur frequently in the context

I eat an orange every day (O) → Co-occur frequently in the context

I eat a car every day (X) → Co-occur rarely in the context

Word2Vec

Co-occurrence probabilities have three important properties

1. Each word has its own unique distribution

2. Similar words have similar distributions

3. Different words have different distributions

Word2Vec

2 (A), (B), (C)의 각 네모 안에서 문맥에 맞는 낱말로 가장 적절한 것은?

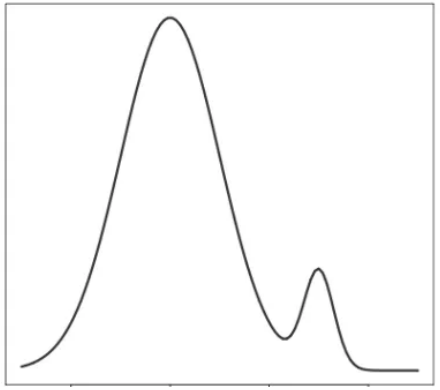
To say that we need to curb anger and our negative thoughts and emotions does not mean that we should deny our feelings. There is an important distinction to be made between denial and restraint. The latter constitutes a (A) [desperate / deliberate] and voluntarily adopted discipline based on an appreciation of the benefits of doing so. This is very different from the case of someone who suppresses emotions such as anger out of a feeling that they need to present a facade of self-control, or out of fear of what others may think. Such behavior is like (B) [healing / closing] a wound which is still infected. We are not talking about rule-following. Where denial and suppression occur, there comes the danger that in doing so the individual stores up anger and resentment. The trouble here is that at some future point they may find they cannot (C) [contain / attain] these feelings any longer.

＊facade 표면, 겉

	(A)		(B)		(C)
①	desperate	……	healing	……	contain
②	desperate	……	healing	……	attain
③	deliberate	……	healing	……	contain
④	deliberate	……	closing	……	contain
⑤	deliberate	……	closing	……	attain

Word2Vec

Word2Vec predicts the co-occurring words



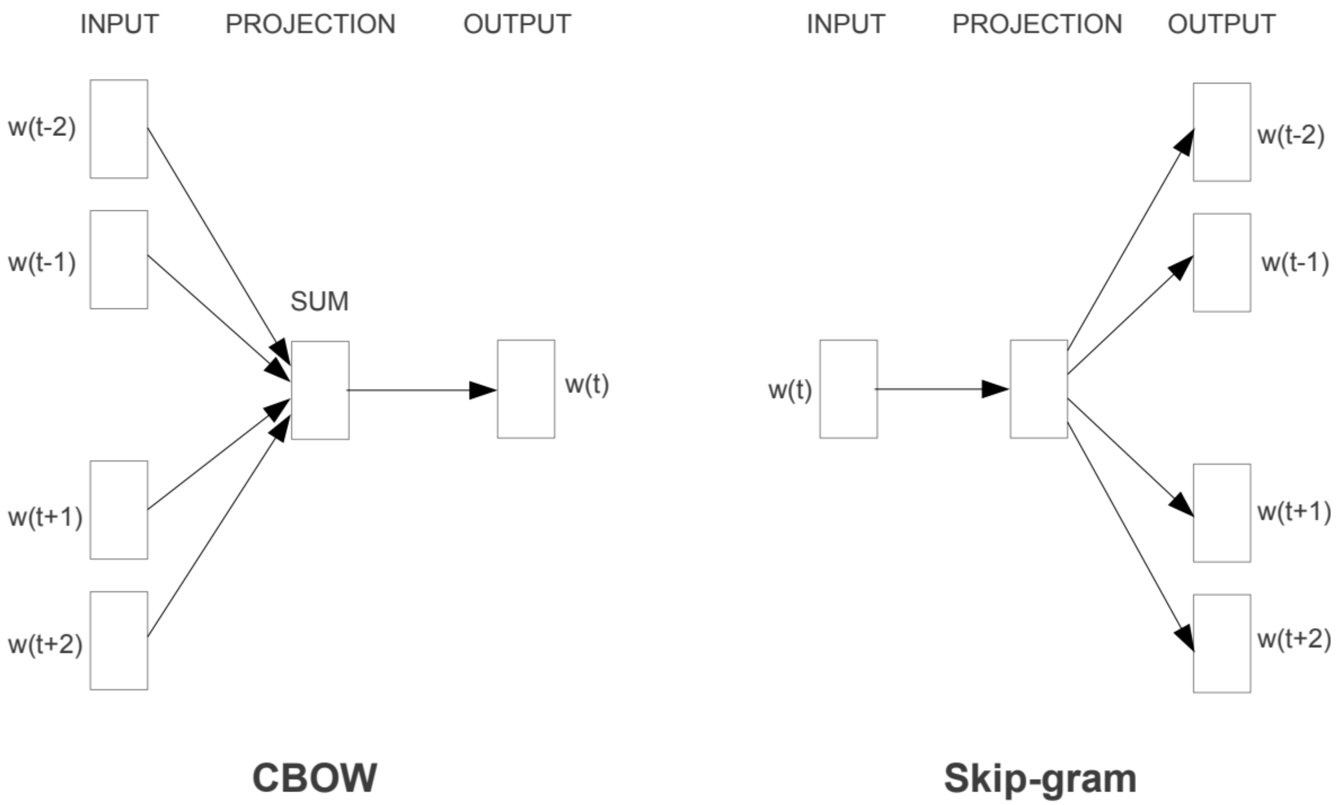
Input → Word Representation → Output

Automatically determined

Word
(one-hot vector)

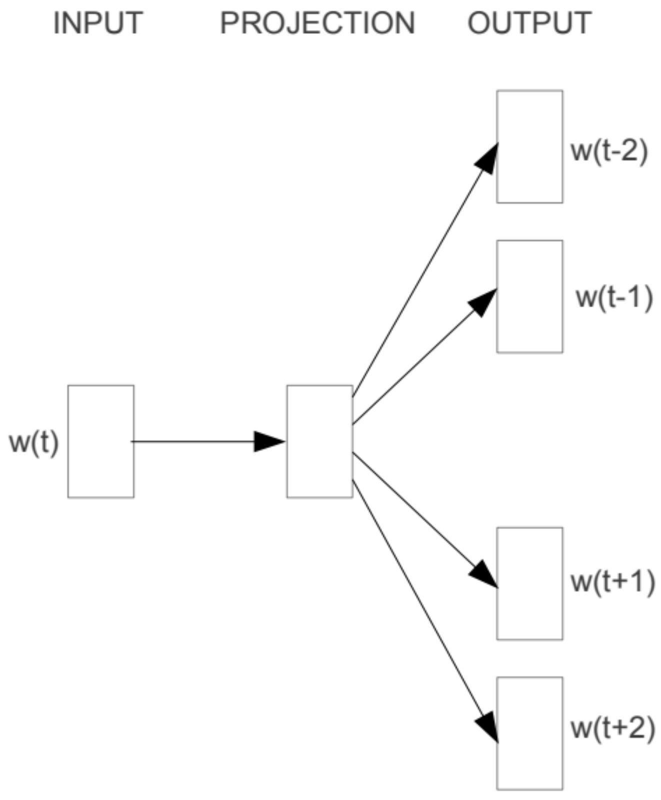
Word
(Predict)

Word2Vec


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
INPUT
PROJECTION
OUTPUT
w(t)
w(t-2)
w(t-1)
w(t+1)
w(t+2)
Skip-gram

Word2Vec

Skip-gram



Skip-gram

Predict context words using a center word

I eat an **orange** every day.

context words context words

Predict

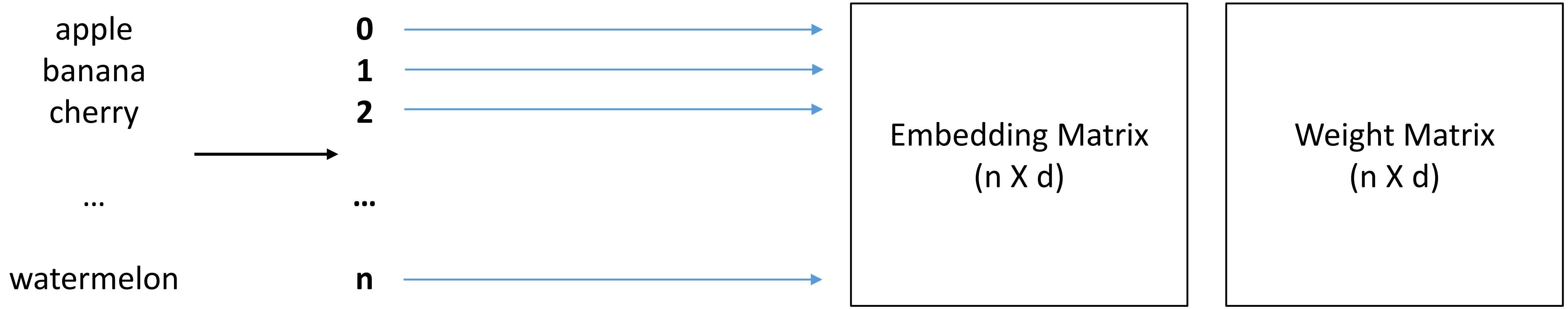
??? ??? ??? **orange** ??? ???

Word2Vec

Skip-gram

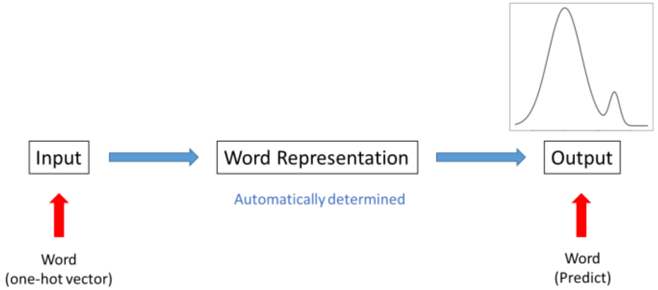
0. Preliminaries

- Build a dictionary and give an index number to each word
- Make two matrices randomly initialized (No bias)



Word2Vec

Skip-gram



1. Word encoding

I eat an **orange** every day.

Where is one-hot vector???

orange

Parameterize

Word vector

from the embedding matrix

Word2Vec

Skip-gram

1. Word encoding

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad \equiv \quad \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

equivalent

Matrix multiplication

Read a row

Which is faster?
Which is easier to implement?

Word2Vec

Skip-gram

1. Word encoding

I eat an **orange** every day.

orange

Parameterize

Word vector

from the embedding matrix

Word2Vec

Skip-gram

2. Predict

Word vector

Probabilities

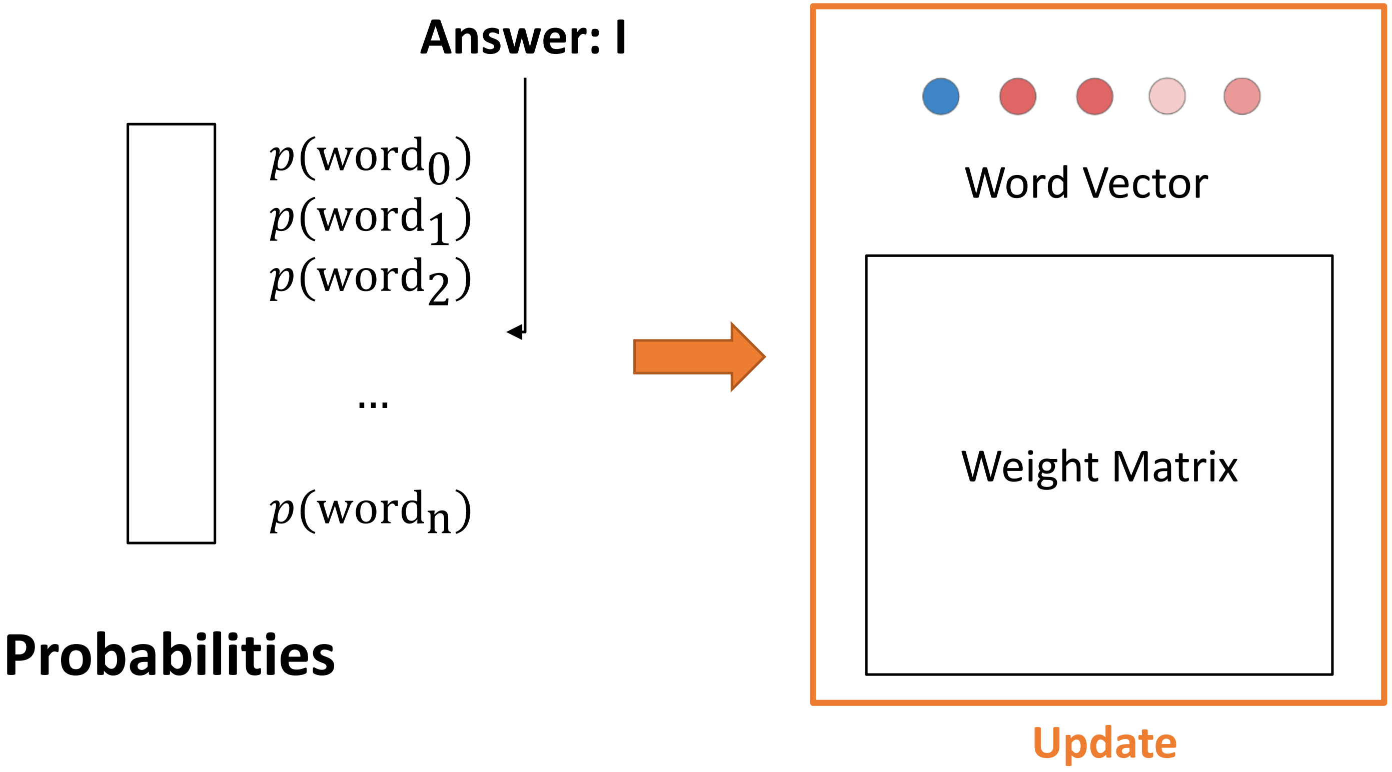
- Each element represent the probability of a word

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.



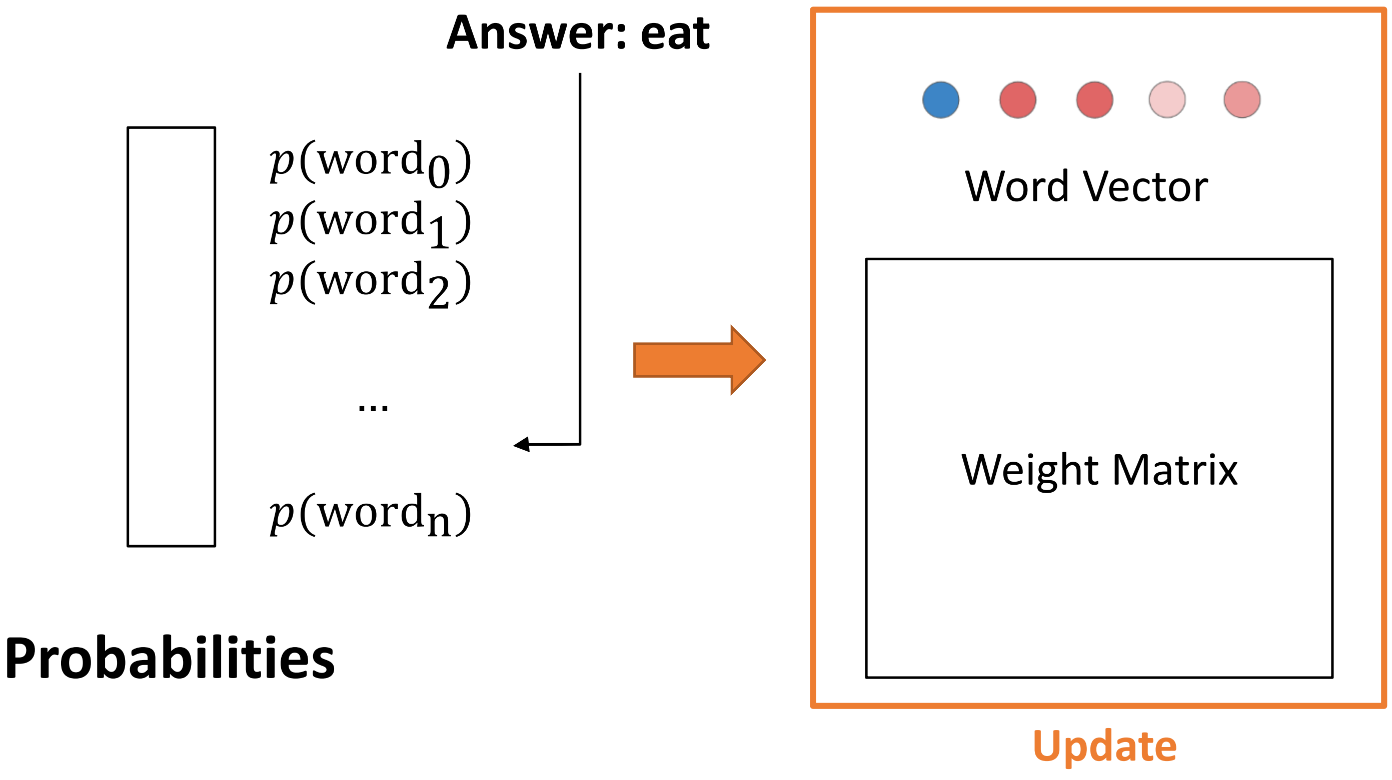
- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.



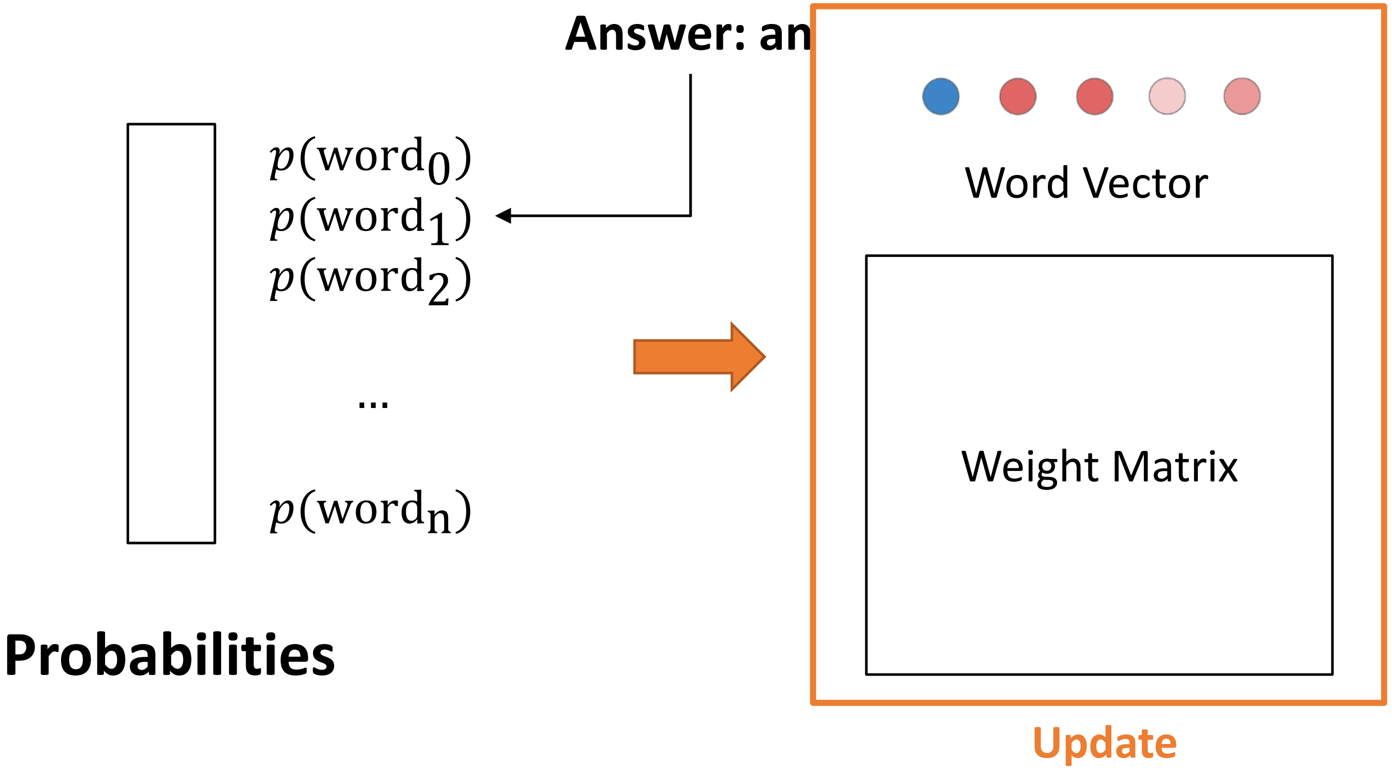
- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.



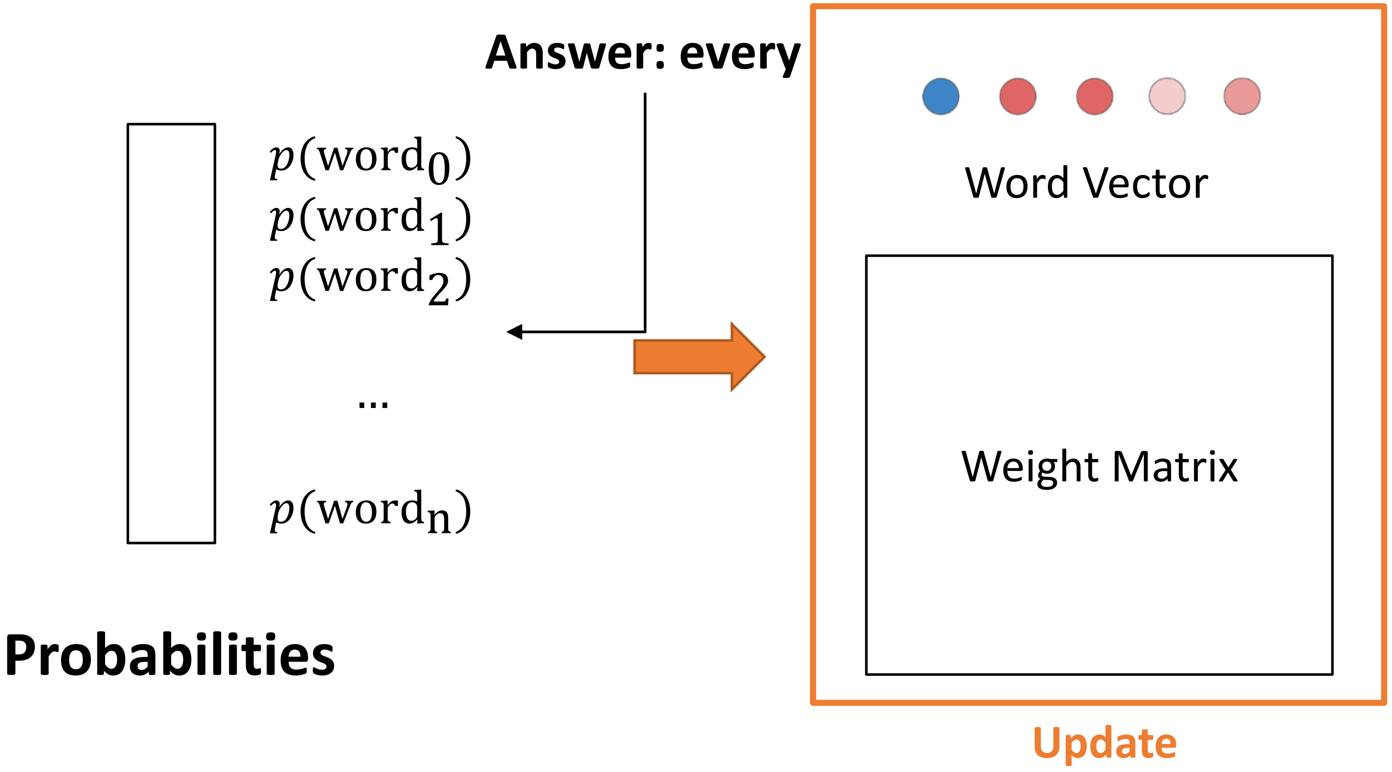
- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

Skip-gram

3. Update

I eat an **orange** every day.



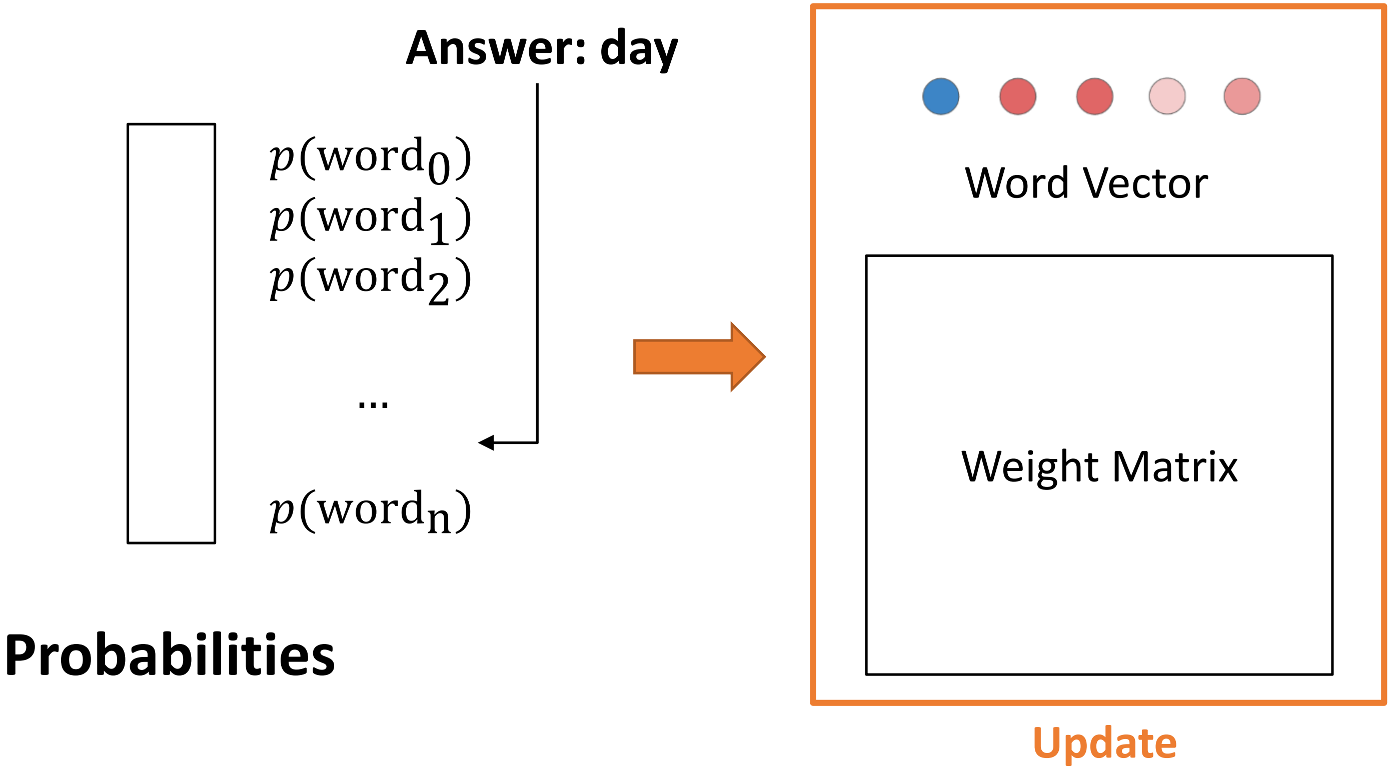
- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

Skip-gram

3. Update

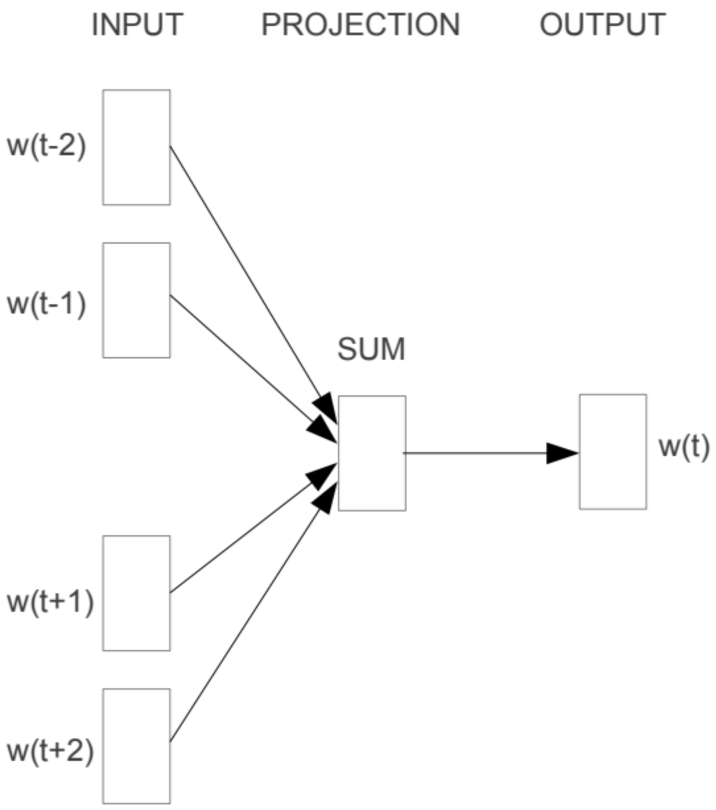
I eat an **orange** every day.



- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

Continuous Bag of Words



CBOW

How frequent the center word occurs in some context?

I eat an **orange** every day.

center word

Predict

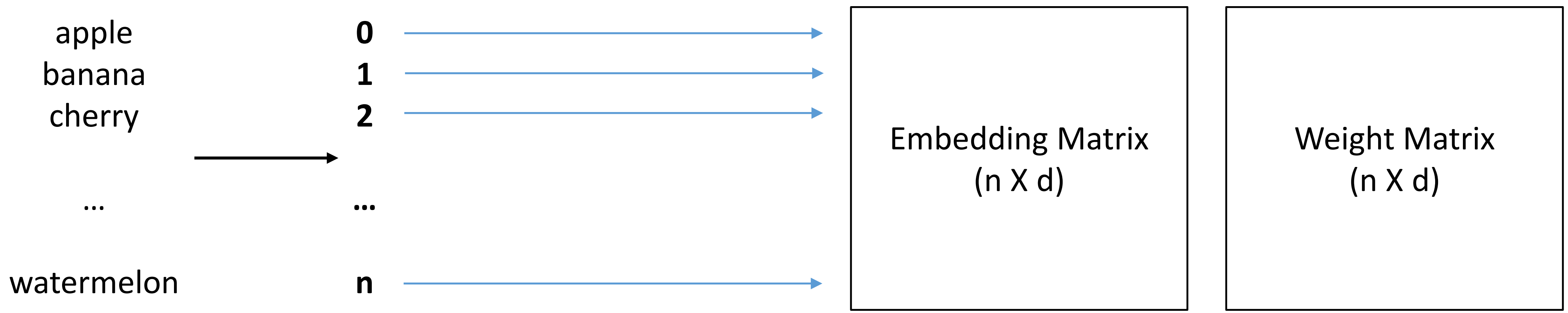
I eat an **???** every day.

Word2Vec

Continuous Bag of Words

0. Preliminaries

- Build a dictionary and give an index number to each word
- Make two matrices randomly initialized (No bias)



Word2Vec

Continuous Bag of Words

1. Context encoding

I eat an $\boxed{\text{???}}$ every day. →

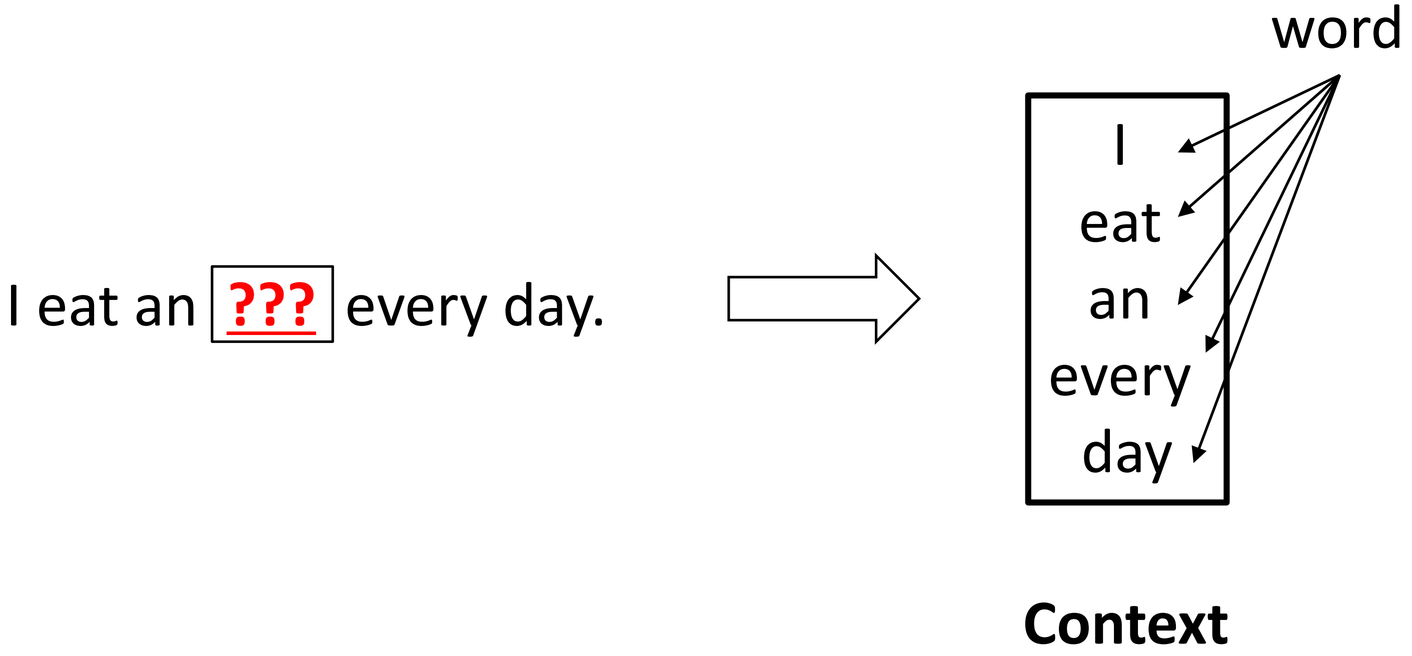
I
eat
an
every
day

Context

Word2Vec

Continuous Bag of Words

1. Context encoding



Word2Vec

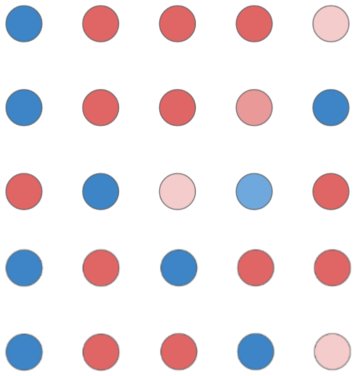
Continuous Bag of Words

1. Context encoding

I eat an ??? every day.

I
eat
an
every
day

Context

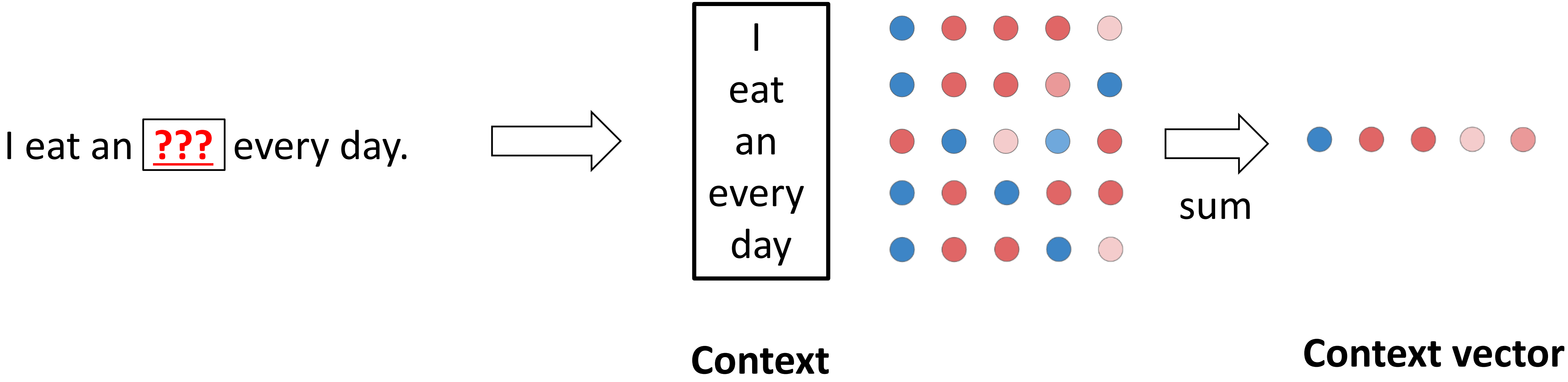


Parameterize

Word2Vec

Continuous Bag of Words

1. Context encoding



Word2Vec

Continuous Bag of Words

2. Prediction

Context vector

Probabilities

- Each element represent probability of a word

Word2Vec

Continuous Bag of Words

3. Update

I eat an **???** every day.

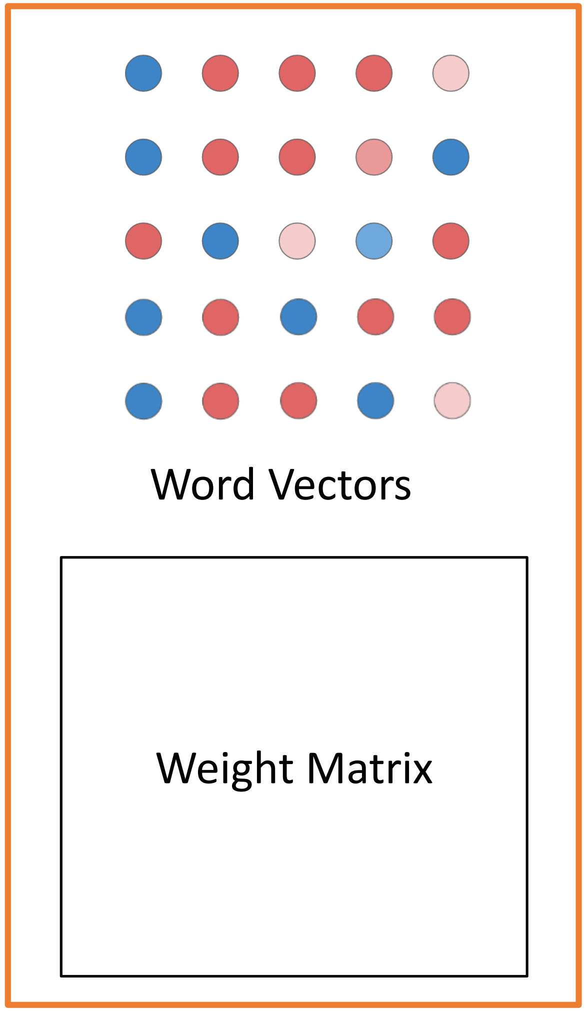
Answer: orange

$p(\mathrm{word}_1)$
$p(\mathrm{word}_2)$
$p(\mathrm{word}_3)$

…

$p(\mathrm{word}_n)$

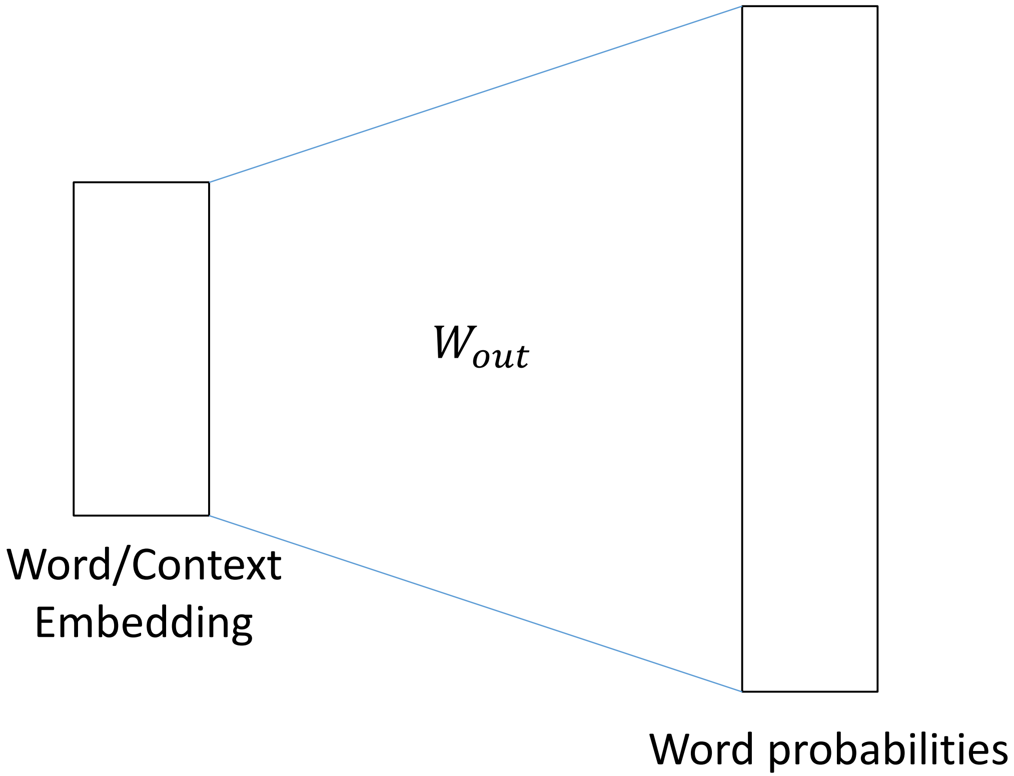
Probabilities



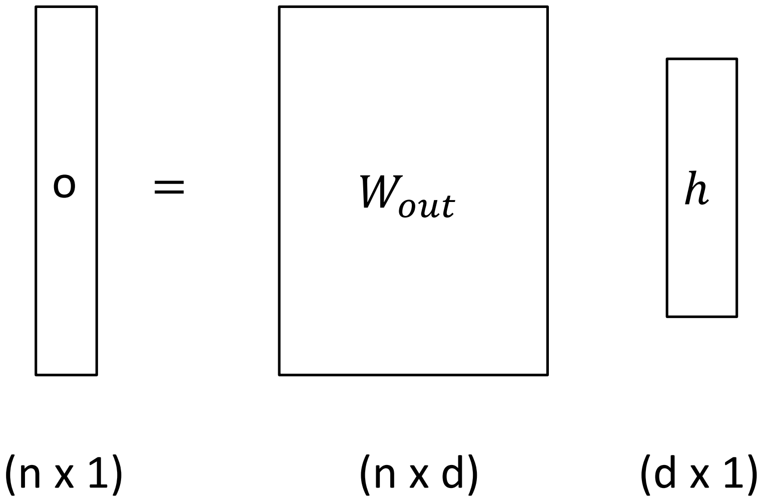
Update

- Negative Log Likelihood Loss
- Backpropagation
- Stochastic Gradient Descent

Word2Vec

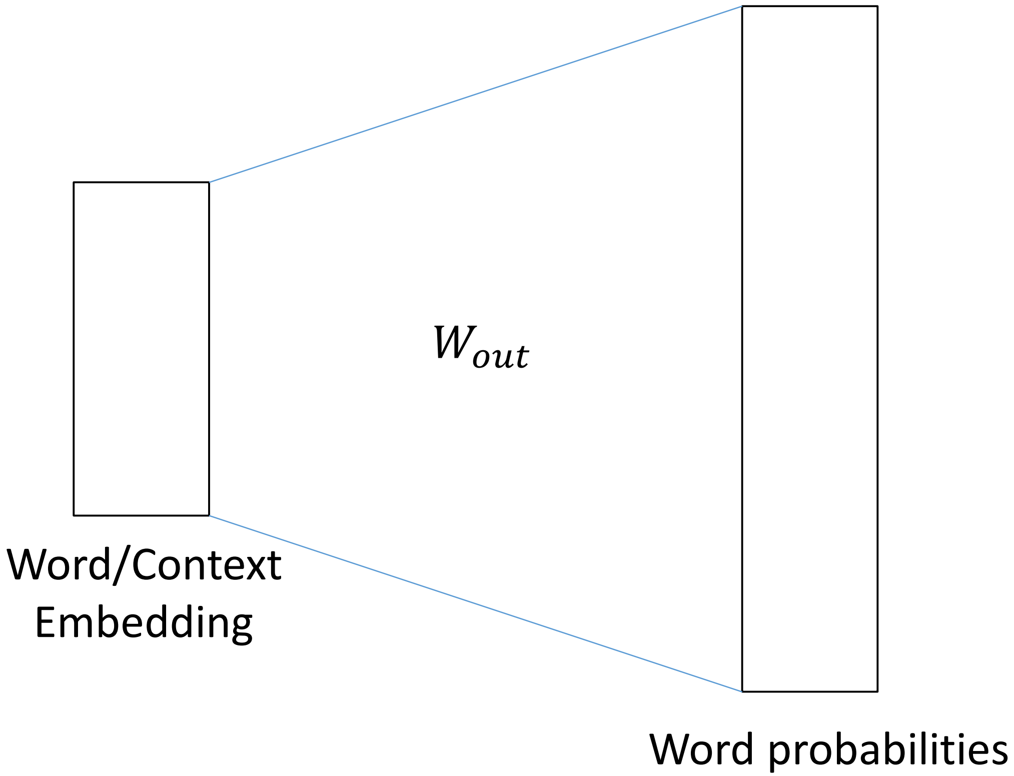


$$y = softmax(W_{out}h)$$



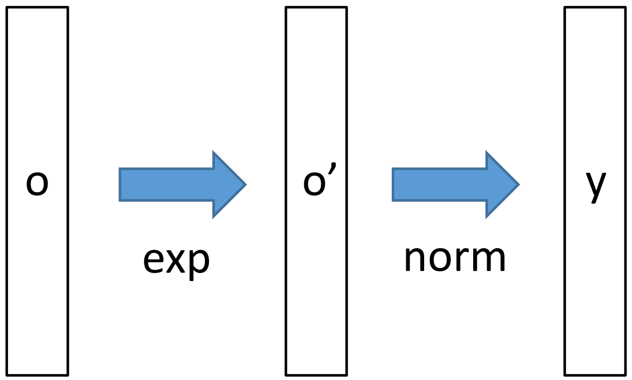
V: vocabulary size
h: embedding dimension

Word2Vec



$$y = softmax(W_{out}h)$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \dots, K.$$



Word2Vec

$$o = W_{out}h$$

$$y = softmax(o) = \frac{e^o}{\sum_k e^k}$$

$$L = NLL(y,t) = -\log(y_t)$$

$$\frac{\partial L}{\partial h} = \boxed{\frac{\partial L}{\partial y}\frac{\partial y}{\partial o}}\frac{\partial o}{\partial h}$$

$$g = \frac{\partial L}{\partial y}\frac{\partial y}{\partial o}$$

$$\frac{\partial L}{\partial W_{out}} = \boxed{\frac{\partial L}{\partial y}\frac{\partial y}{\partial o}}\frac{\partial o}{\partial W_{out}}$$

$$g_k = \begin{cases} y_k - 1 & (k = t) \\ y_k & (k \neq t) \end{cases}$$

Word2Vec

$o = W_{out}h$

$y = softmax(o) = \frac{e^o}{\sum_k e^k}$

$L = NLL(y, t) = -\log(y_t)$

$$\frac{\partial L}{\partial h} = W_{out}\, g$$

$$\frac{\partial L}{\partial W_{out}} = hg^T$$

Word2Vec

$$o = W_{out} h$$

$$y = softmax(o) = \frac{e^o}{\sum_k e^k}$$

$$L = NLL(y, t) = -\log(y_t)$$

$$\frac{\partial L}{\partial h} = W_{out}\, g$$

$$\frac{\partial L}{\partial W_{out}} = h g^T$$

CBOW

$$h = w_a + w_b + w_c + w_d$$

$$\frac{\partial h}{\partial w_a}, \frac{\partial h}{\partial w_b}, \frac{\partial h}{\partial w_c}, \frac{\partial h}{\partial w_d} = 1$$

$$w_a = w_a - \eta \frac{\partial L}{\partial h}$$

$$w_b = w_b - \eta \frac{\partial L}{\partial h}$$

$$w_c = w_c - \eta \frac{\partial L}{\partial h}$$

$$w_d = w_d - \eta \frac{\partial L}{\partial h}$$

$$W_{out} = W_{out} - \eta \frac{\partial L}{\partial W_{out}}$$

Skip-gram

$$h = w_k$$

$$w_k = w_k - \eta \frac{\partial L}{\partial h}$$

$$W_{out} = W_{out} - \eta \frac{\partial L}{\partial W_{out}}$$

Word2Vec

CBOW vs Skip-gram

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

Word2Vec

Better and Faster

Model Architecture	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness Test Set [20]
	Semantic Accuracy [%]	Syntactic Accuracy [%]	
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

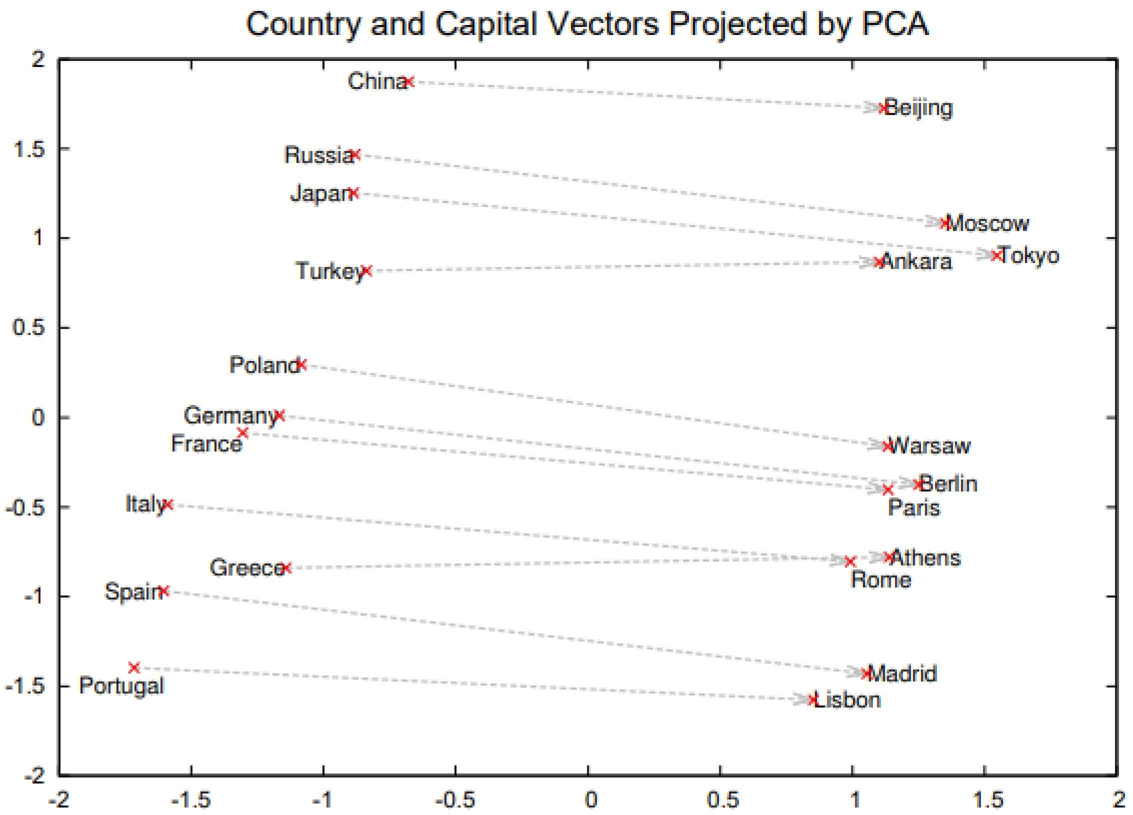
Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

Word2Vec

Additive Compositionality

vec("Paris") - vec("France")

= vec("Berlin") - vec("Germany")



Assignment 4

- Word2Vec Implementation
  - CBOW and Skip-gram
    - Forward path
    - Backward path
    - Return : cost value and gradient of two word vectors

Assignment 4

- Word2Vec Implementation

```python
def Skipgram(center, context, inputMatrix, outputMatrix):
################################  Input  ################################
# center : Index of a centerword (type:int)                               #
# context : Index of a contextword (type:int)                             #
# inputMatrix : Weight matrix of input (type:torch.tesnor(V,D))           #
# outputMatrix : Weight matrix of output (type:torch.tesnor(V,D))         #
#########################################################################

###############################  Output  ################################
# loss : Loss value (type:torch.tensor(1))                                #
# grad_emb : Gradient of word vector (type:torch.tensor(1,D))             #
# grad_out : Gradient of outputMatrix (type:torch.tesnor(V,D))            #
#########################################################################

    loss = None
    grad_emb = None
    grad_out = None

    return loss, grad_emb, grad_out

def CBOW(center, context, inputMatrix, outputMatrix):
################################  Input  ################################
# center : Index of a centerword (type:int)                               #
# context : Indices of contextwords (type:list(int))                      #
# inputMatrix : Weight matrix of input (type:torch.tesnor(V,D))           #
# outputMatrix : Weight matrix of output (type:torch.tesnor(V,D))         #
#########################################################################

###############################  Output  ################################
# loss : Loss value (type:torch.tensor(1))                                #
# grad_emb : Gradient of word embedding (type:torch.tensor(1,D))          #
# grad_out : Gradient of outputMatrix (type:torch.tesnor(V,D))            #
#########################################################################

    loss = None
    grad_emb = None
    grad_out = None

    return loss, grad_emb, grad_out
```

Assignment 4

- Word2Vec Impleme

```python
def word2vec_trainer(corpus, word2ind, mode="CBOW", dimension=64, learning_rate=0.05, iteration=50000):

    #initialization
    W_emb = torch.randn(len(word2ind), dimension) / (dimension**0.5)
    W_out = torch.randn(len(word2ind), dimension) / (dimension**0.5)
    window_size = 5


    losses=[]
    for i in range(iteration):
        #Training word2vec using SGD
        centerWord, contextWords = getRandomContext(corpus, window_size)
        centerInd = None
        contextInds = None

        #learning rate decay
        lr = learning_rate*(1-i/iteration)

        if mode=="CBOW":
            L, G_emb, G_out = CBOW(centerInd, contextInds, W_emb, W_out)
            W_emb[contextInds] -= lr*G_emb
            W_out -= lr*G_out
            losses.append(L.item())

        elif mode=="SG":
            for contextInd in contextInds:
                L, G_emb, G_out = Skipgram(centerInd, contextInd, W_emb, W_out)
                W_emb[centerInd] -= lr*G_emb.squeeze()
                W_out -= lr*G_out
                losses.append(L.item())

        else:
            print("Unkwnown mode : "+mode)
            exit()

        if i%10000==0:
            avg_loss=sum(losses)/len(losses)
            print("Loss : %f" %(avg_loss,))
            losses=[]

    return W_emb, W_out
```

Assignment 4

- Word2Vec Experiment

Analogical reasoning task

"work" : "works" :: "speak" : ?

z =vec("works") - vec("work") + vec("speak")

Find 5 words whose vector is similar to z
(cosine similarity)

*text8 only includes lower cases
**Exclude question words from the predictions

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space", ICLR 2013

Assignment 4

- Word2Vec Experiment
  - In this assignment, 9 types are used

Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Total 36 questions

work :: works = speak :: speaks

- works – work + speak
- work – works + speaks
- speaks – speak + work
- speak – speaks + works

Report top 5 accuracy
Any of 5 predictions is correct -> correct
None of 5 predictions is correct -> wrong

Submission

- Due date : ~5/20(수) 23:59
- Submission : Online submission on blackboard
- word2vec.py + Report(pdf)
- Report should include
    1. Each member's contribution
    2. Explanation of your code
    3. Analysis of experiments
- You must implement the components yourself!
- You must specify each member's contribution (role) in this assignment
- File name : TeamID_word2vec.zip

Q&A

- Data intelligence lab.
- irish07@korea.ac.kr (박준형)