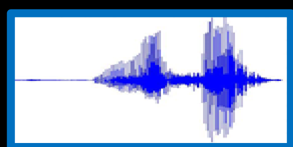
Deep Learning for Language Modeling

Hung-yi Lee

Language model

Speech Recognition



X → Speech Recognition → 天氣真好

$\tilde{W}$

$$\tilde{W} = arg \max_{W} P(W|X)$$

$$= arg \max_{W} \frac{P(X|W)P(W)}{P(X)}$$

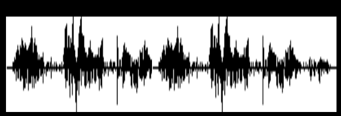
$$= arg \max_{W} P(X|W)P(W)$$

$P(X|W)$:
Acoustic Model

$P(W)$:
Language Model

Language model

- Language model: Estimated the probability of word sequence
  - Word sequence: $w_1, w_2, w_3, \ldots, w_n$
  - $P(w_1, w_2, w_3, \ldots, w_n)$
- Useful in speech recognition
  - Different word sequence can have the same pronunciation



recognize speech
or
wreck a nice beach

If P(recognize speech)
>P(wreck a nice beach)

Output =
“recognize speech”

Language model

P("wreck a nice beach")
=P(wreck|START)P(a|wreck)
P(nice|a)P(beach|nice)

- How to estimate $P(w_1, w_2, w_3, \ldots, w_n)$
- Collect a large amount of text data as training data
  - However, the word sequence $w_1, w_2, \ldots, w_n$ may not appear in the training data
- N-gram language model: $P(w_1, w_2, w_3, \ldots, w_n) = P(w_1|START)P(w_2|w_1) \ldots P(w_n|w_{n-1})$
- Estimate P(beach|nice) from training data

$$P(\text{beach}|\text{nice}) = \frac{C(nice\ beach)}{C(nice)}$$

Count of "nice beach" in the training data

Count of "nice" in the training data

Language model - Smoothing

- Training data:
  - The dog ran ……
  - The cat jumped ……

This is called **language model smoothing**.

P(jumped | dog) = ~~0~~ 0.0001

P(ran | cat) = ~~0~~ 0.0001

Give some small probability

➢ The probability is not accurate.

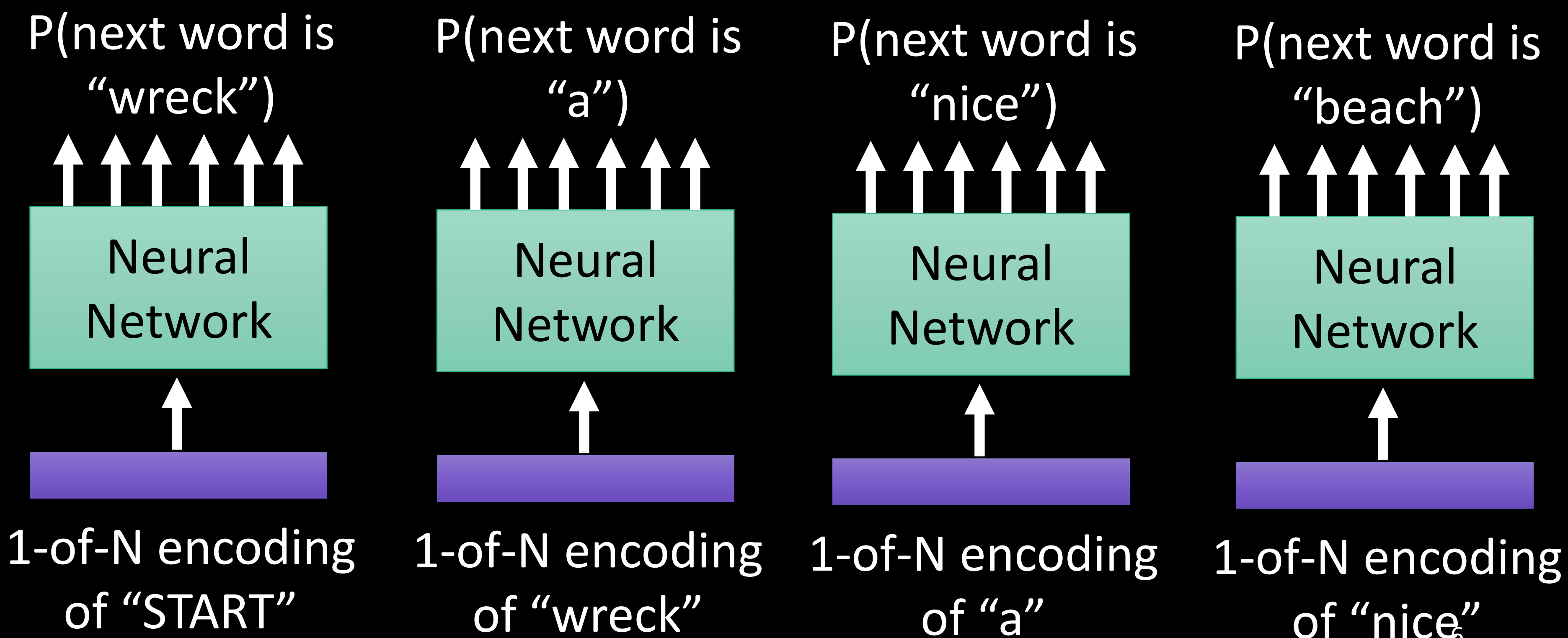
➢ The phenomenon happens because we cannot collect all the possible text in the world as training data.

Neural-network based LM

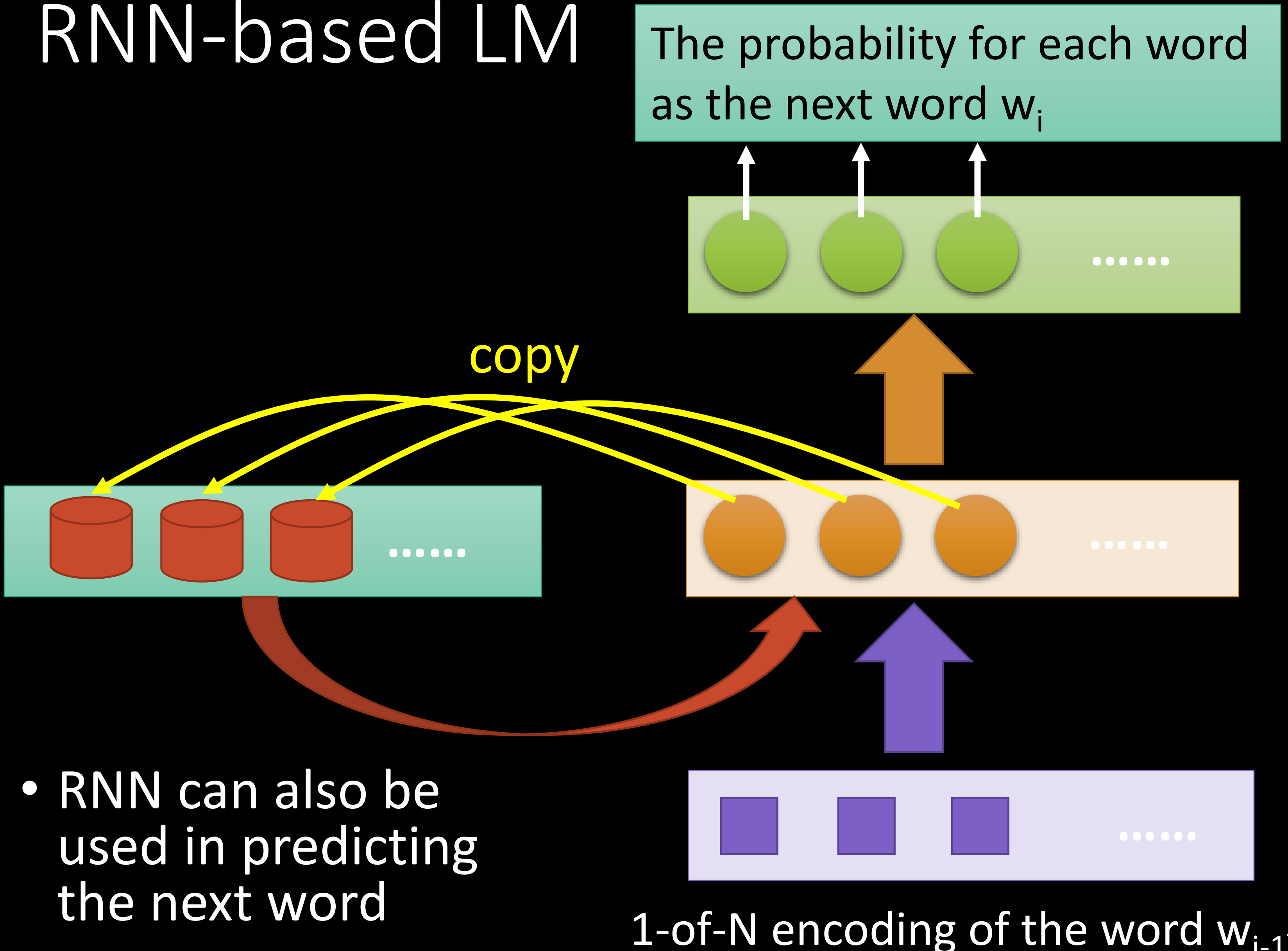
P("wreck a nice beach")

=P(wreck|START)P(a|wreck)P(nice|a)P(beach|nice)

P(b|a): not from counting, using NN to predict the next word.

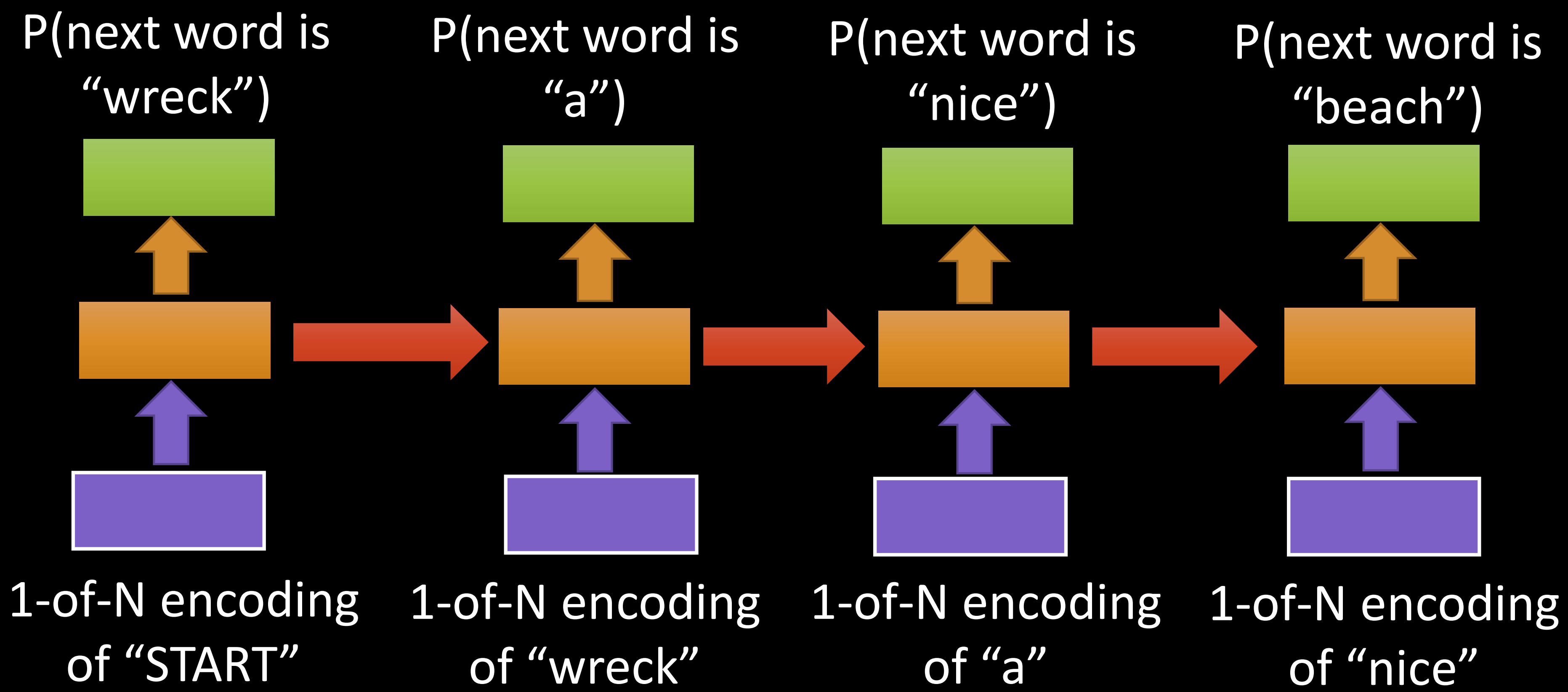


RNN-based LM



- RNN can also be used in predicting the next word

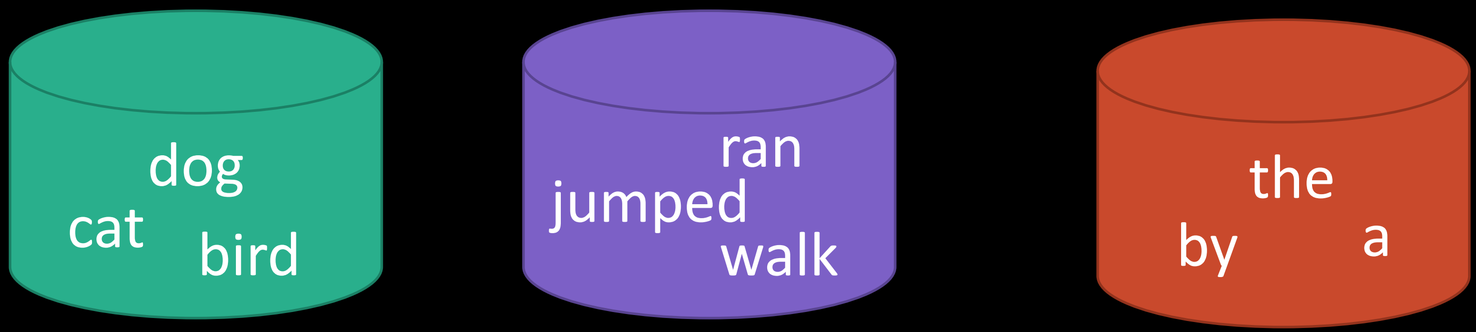
RNN-based LM



- Model long-term information
- People also used Deep RNN or LSTM

Why RNN works? – Class-based

class 1: ***A***nimal class 2: ***V***erb class 3: ***F***unction word

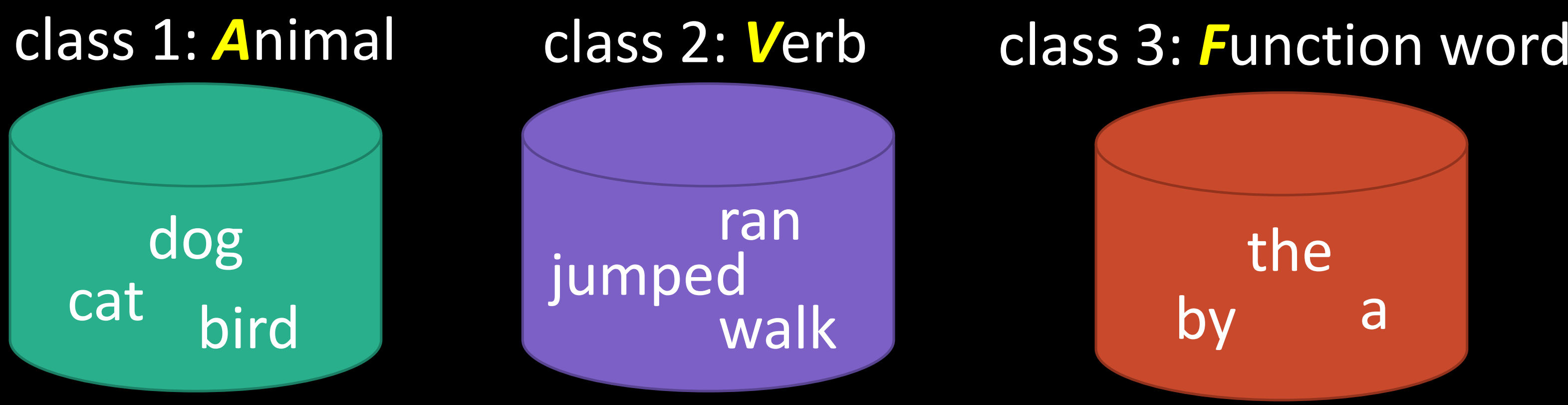


$W = \text{“}w_1\ w_2\ w_3\text{”}$ $C(w_i)$: class of word w_i

$$\cancel{P(W) = P(w_1|START)\,P(w_2|w_1)\,P(w_3|w_2)}$$

$$P(W) = P(C(w_1)|START)\,P(C(w_2)|C(w_1))\,P(C(w_3)|C(w_2))$$
$$\times\ P(w_1|C(w_1))\,P(w_2|C(w_2))\,P(w_3|C(w_3))$$

Why RNN works? – Class-based



W = “the dog ran”

F *A* *V*

P(W) = P(*F*|START) P(*A*|*F*) P(*V*|*A*)

X P(the|*F*) P(dog|*A*) P(ran|*V*)

P(class i | class j) and P(word w| class i) are estimated from training data.

Why RNN works? – Class-based

P(class i | class j) and P(word w| class i) are estimated from training data.

Training data

the	dog	ran
F	***A***	***V***

the	cat	jumped
F	***A***	V

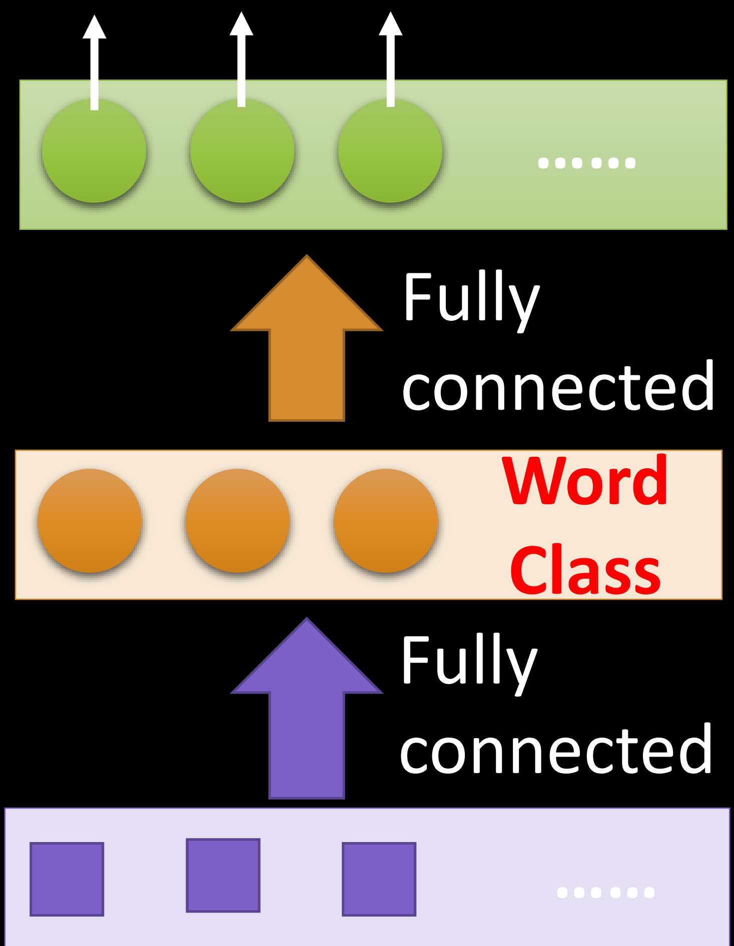
W = “the cat ran”

F ***A*** ***V***

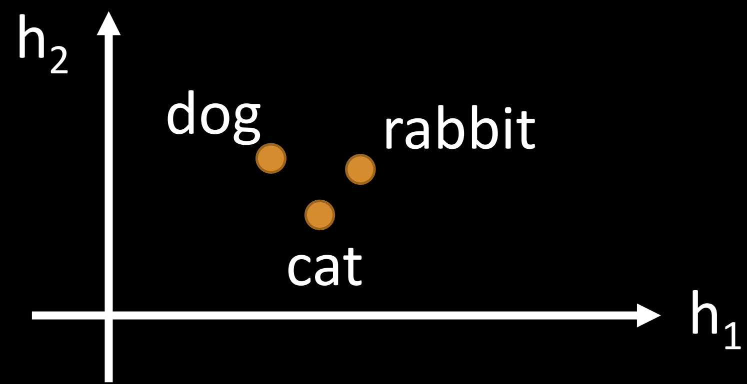
P(ran|cat) is zero given the training data

However, P(***V***erb | ***A***nimal) is not zero

Why RNN works? – Class-based



The hidden layer of the related words are close.



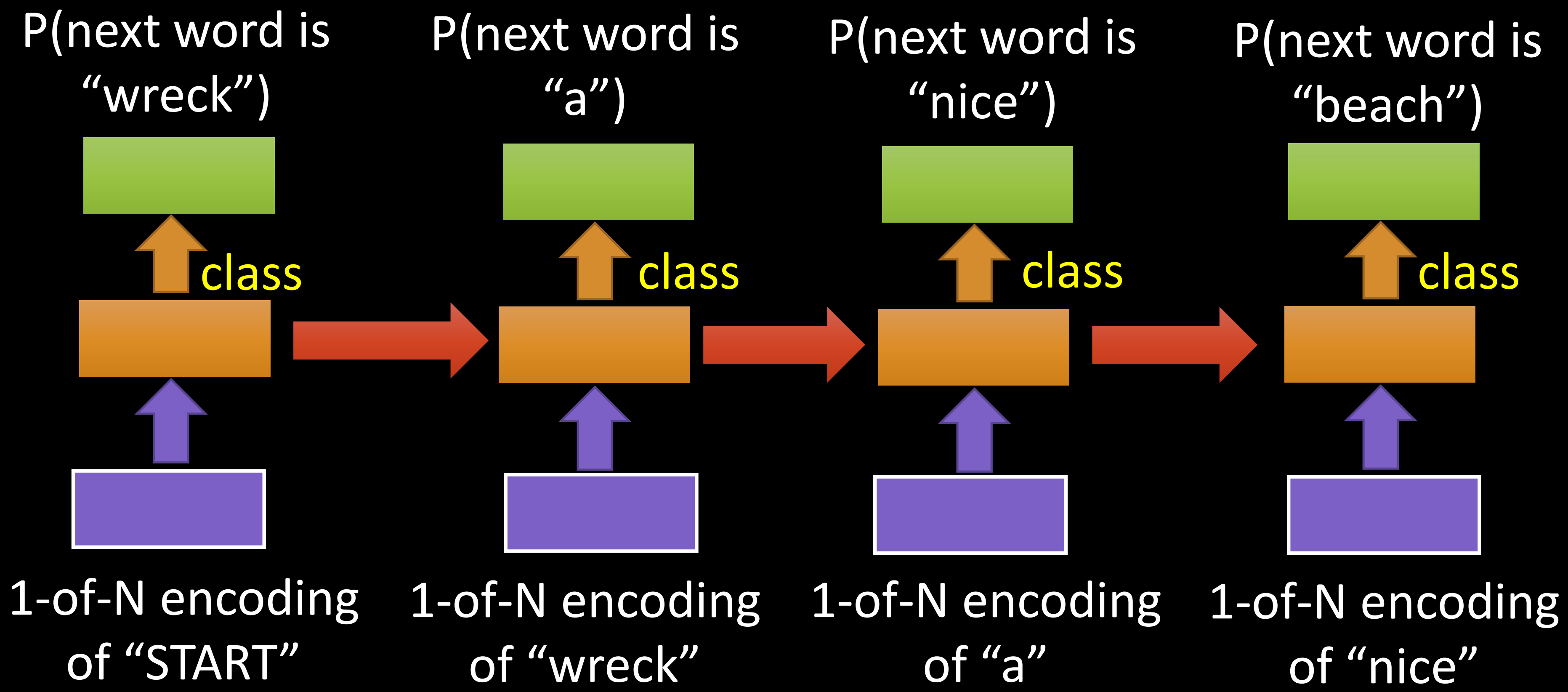
If P(jump|dog) is large, then P(jump|cat) increase accordingly.

(even there is not "… cat jump …" in the data)

Smoothing is automatically done.

Why RNN works? – Class-based

- Like class-based LM



Large Output Layer ……

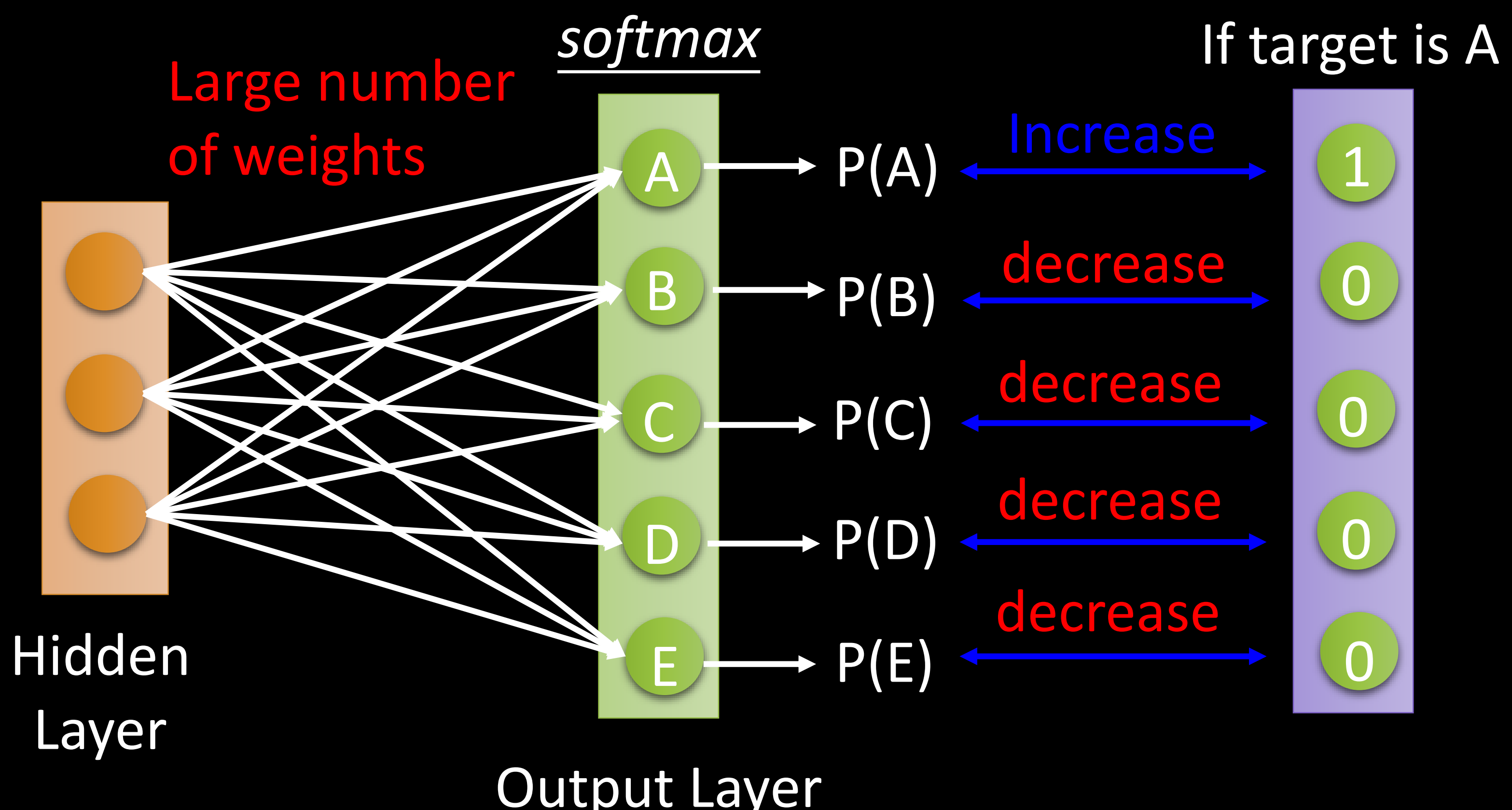
- Factorization of the Output Layer
  - Ref: Mikolov Tomáš: Statistical Language Models based on Neural Networks. PhD thesis, Brno University of Technology, 2012. (chapter 3.4.2)
- Noise Contrastive Estimation (NCE)
  - Ref: Ref: X. Chen, X. Liu, M. J. F. Gales and P. C. Woodland, "Recurrent neural network language model training with noise contrastive estimation for speech recognition,“ ICASSP, 2015
- More ways to deal with the large output layer
  - Hinton’s course on Coursera: Ways to Deal with the Large Number of Possible Outputs
  - https://www.youtube.com/watch?v=vLmgSo9LVMk

Appendix

Some ways to deal with the large output layer

Large Output Layer

Normal Training

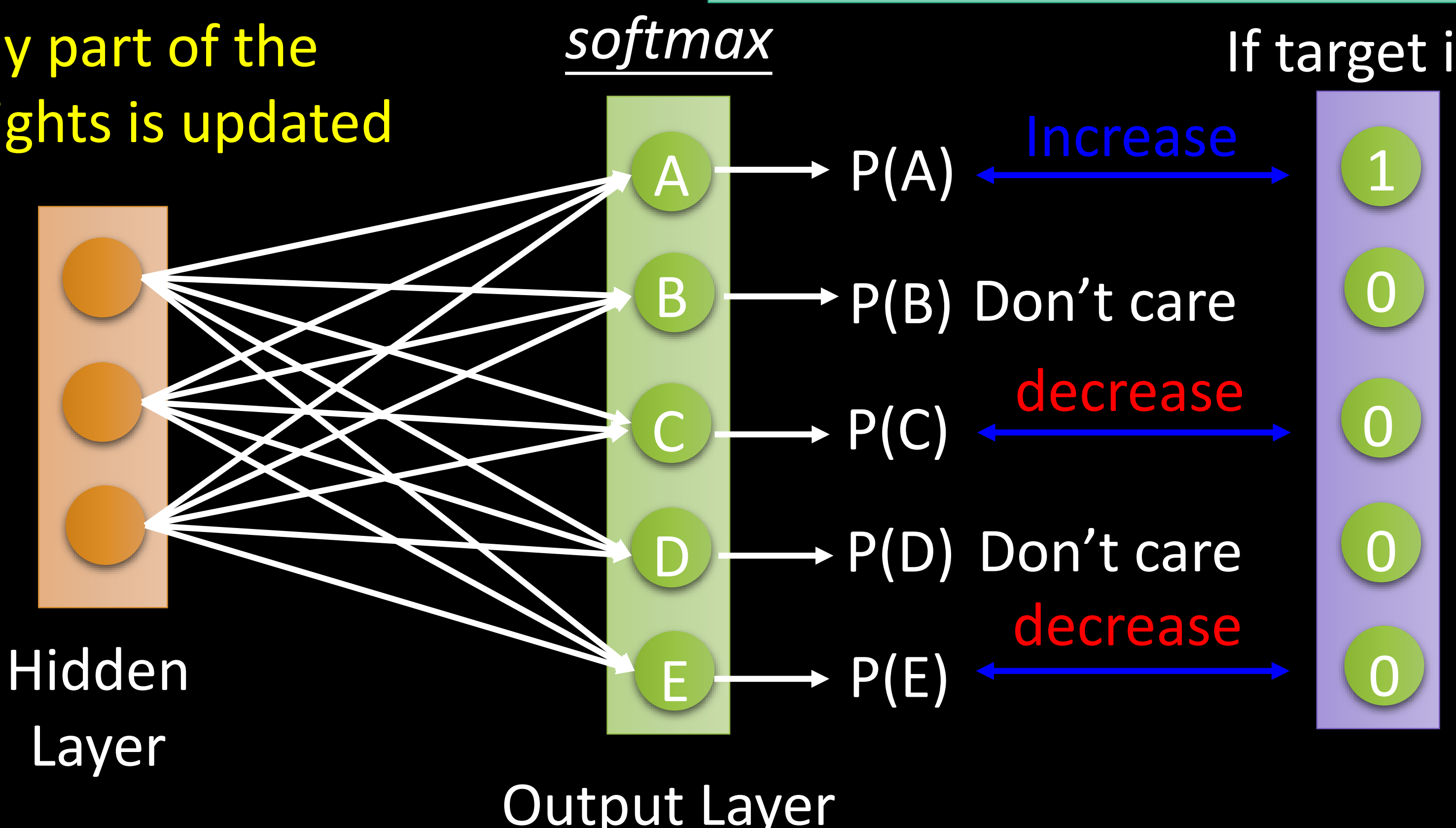


Large Output Layer – Noise Contrastive Estimation (NCE)

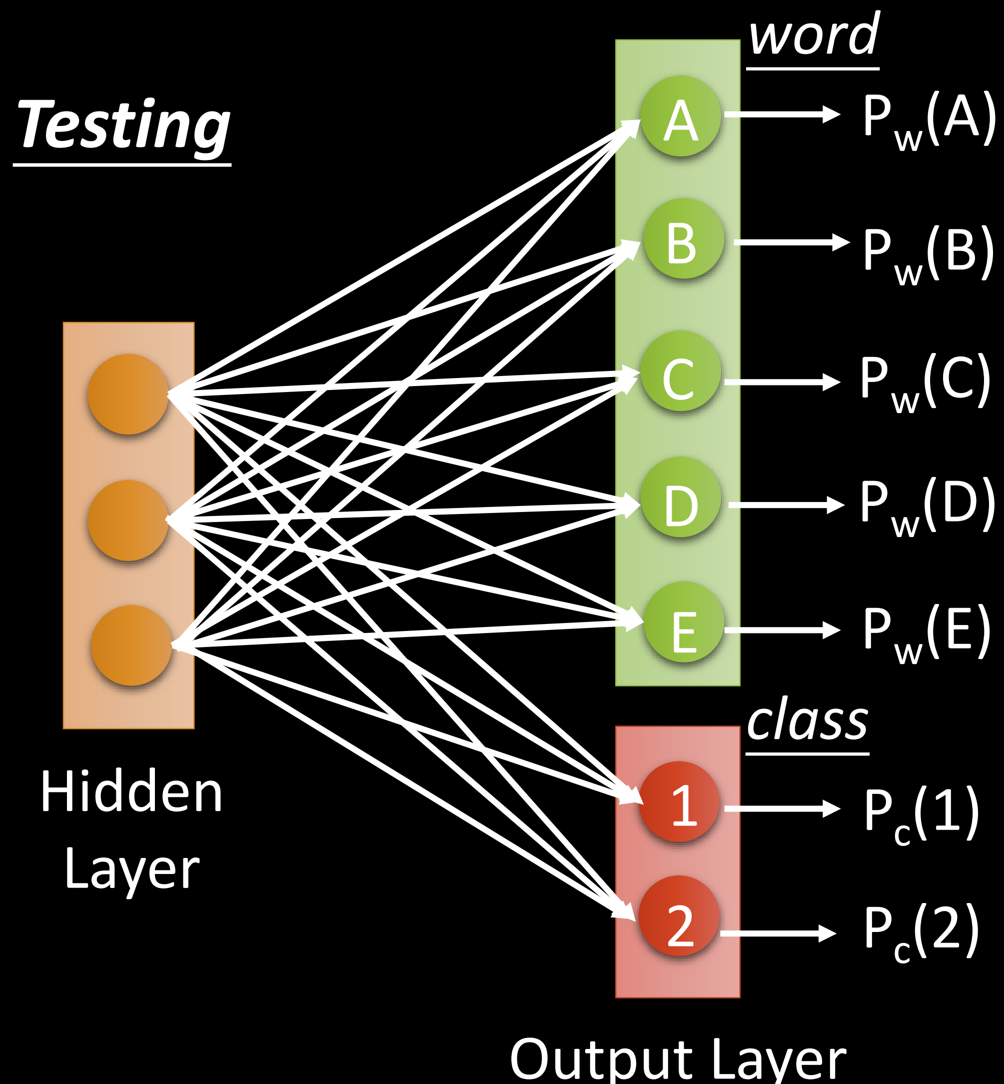
Randomly sample some words to suppress the probability

NCE

Only part of the weights is updated



Large Output Layer - Factorization



Word	Class
A	1
B	1
C	1
D	2
E	2

$$P(A) = P_w(A) \text{ X } P_c(1)$$

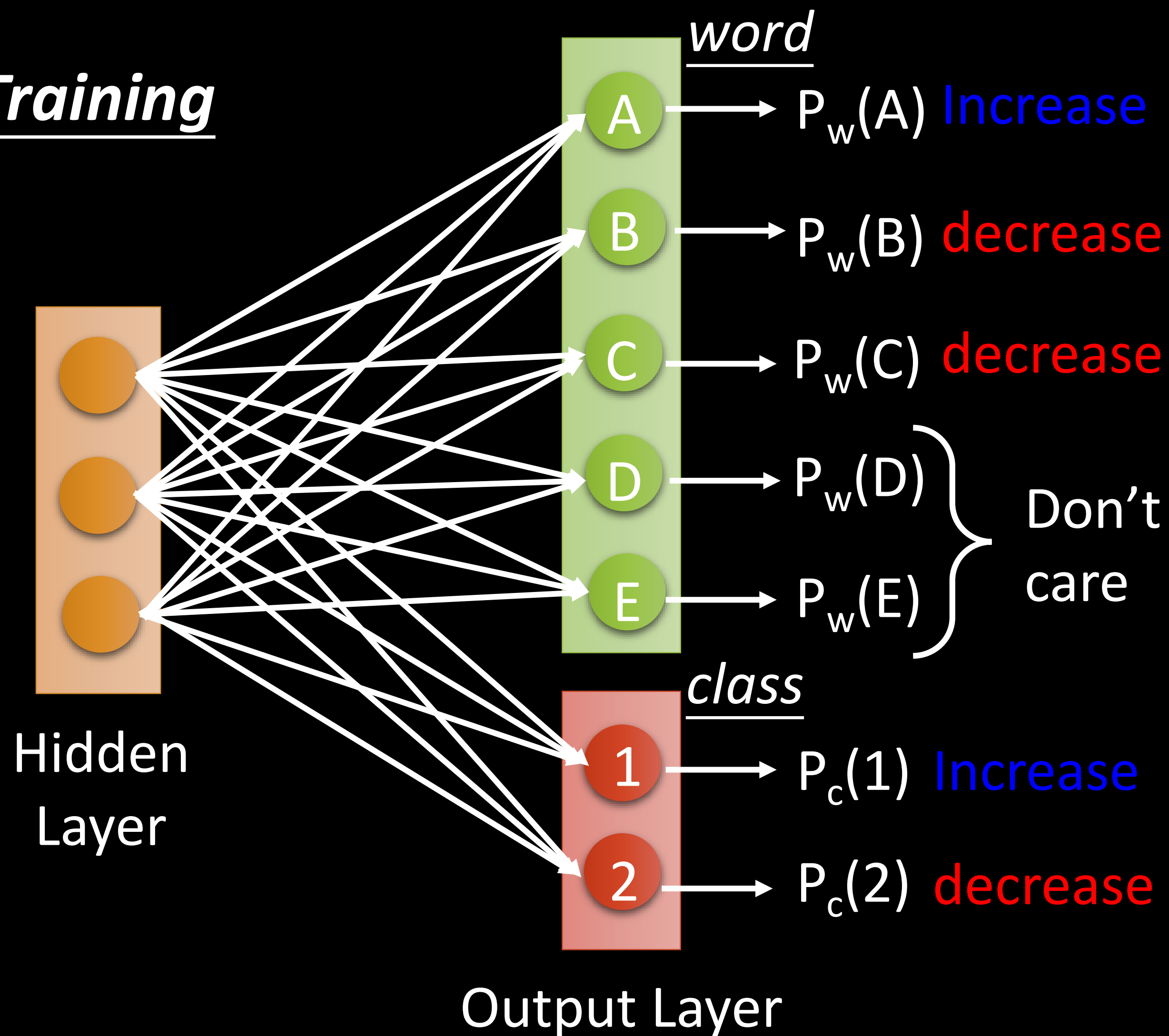
……

$$P(D) = P_w(D) \text{ X } P_c(2)$$

……

Large Output Layer - Factorization

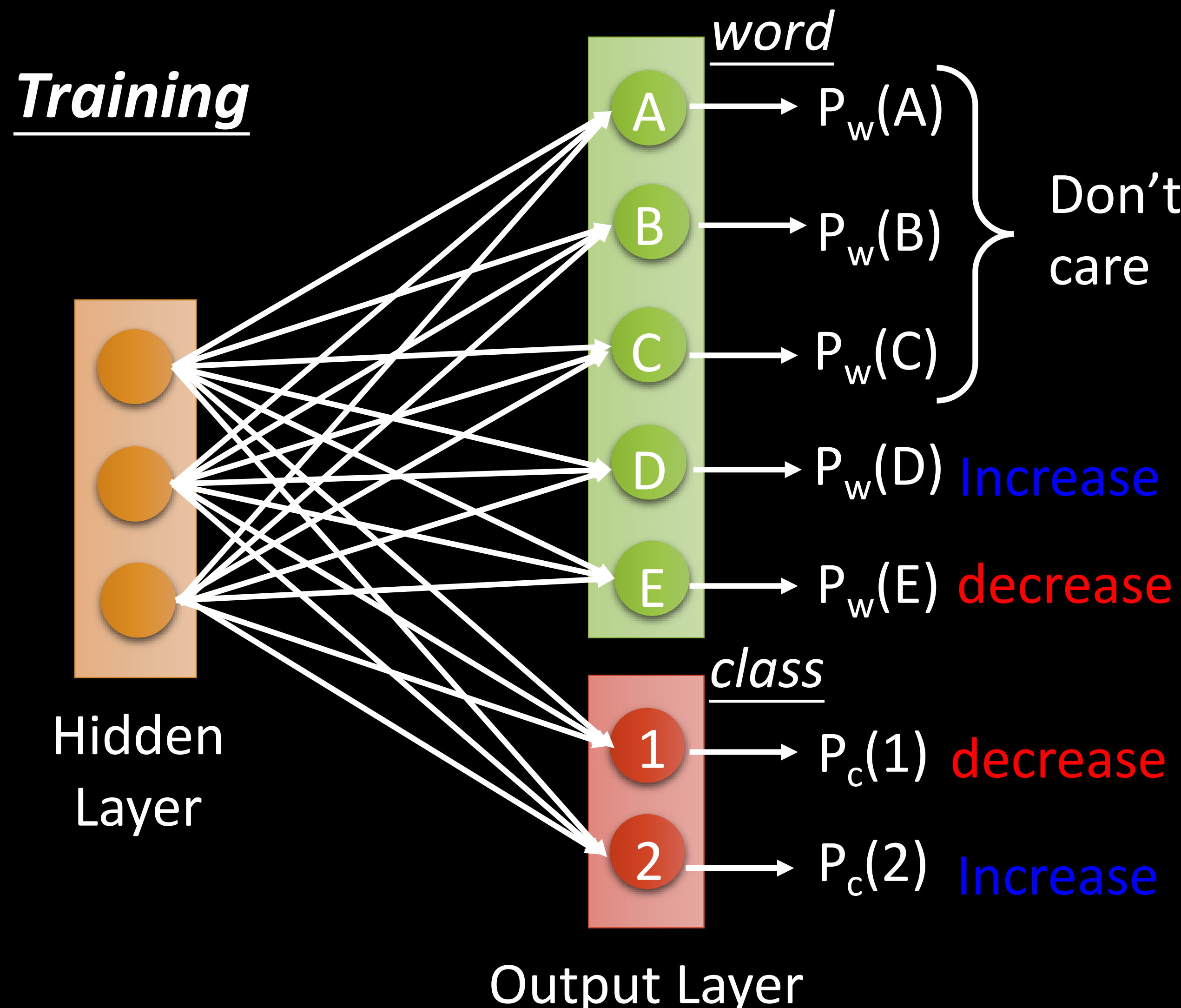
Training



If target is A

A belongs to class 1

Large Output Layer - Factorization



If target is D

D belongs to class 2

Sentence Completion Task

- https://docs.google.com/presentation/d/1K3nMJsCzbdNTvp3GslgX59-drtVYJ558Lya2DXOeqeU/edit