Neural Networks and Neural Language Models

Y. ZHAO, Reading Group Seminar, Aizawa-lab

20180601

Outline

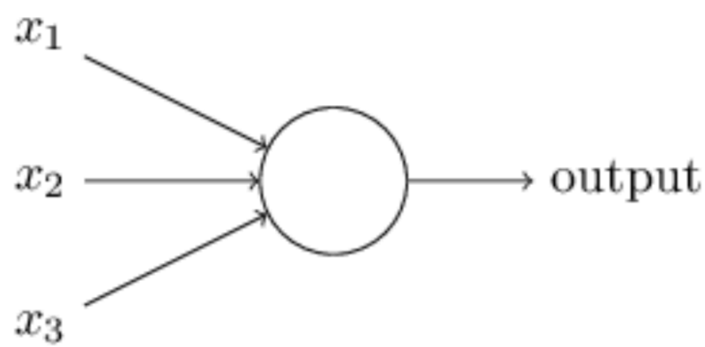
- Single Neuron
- Multiple Neurons (Neural Network)
- Neural Language Model
- Summary

Outline

- Single Neuron
- Multiple Neurons (Neural Network)
- Neural Language Model
- Summary

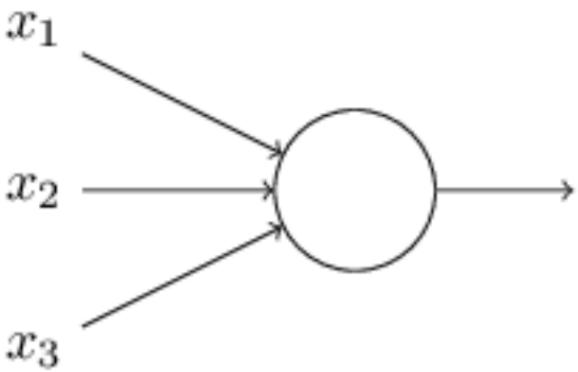
Perceptrons (Neuron)

- Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts (1943).



- Rosenblatt introduced weights, w1,w2,…, real numbers expressing the importance of the respective inputs to the output.
- The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum w_i x_i$ is less than or greater than some *threshold value (b)*.

Perceptrons (Neuron)

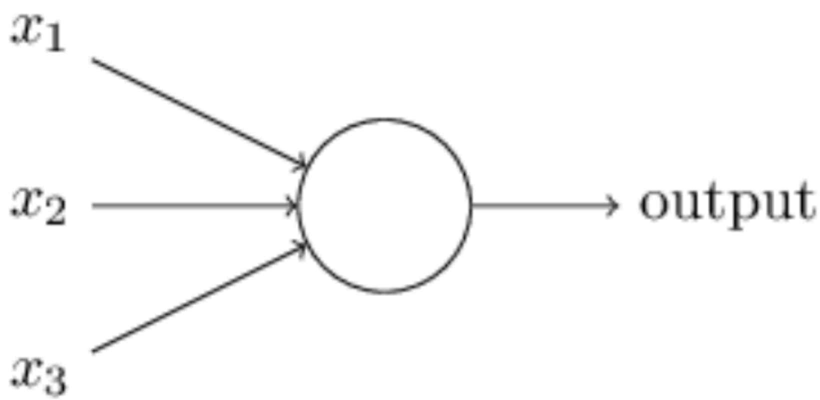


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold (-b)} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold (-b)} \end{cases}$$

That's how a perceptron (Neuron) works! the perceptron is a device that makes decisions by weighing up **evidences (like $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$)**.

We can manually set the parameters w_1, w_2, w_3 and b to have a decision-making strategy. Even better, these weights can be learned from data.

Weakness of Perceptrons (Neuron)



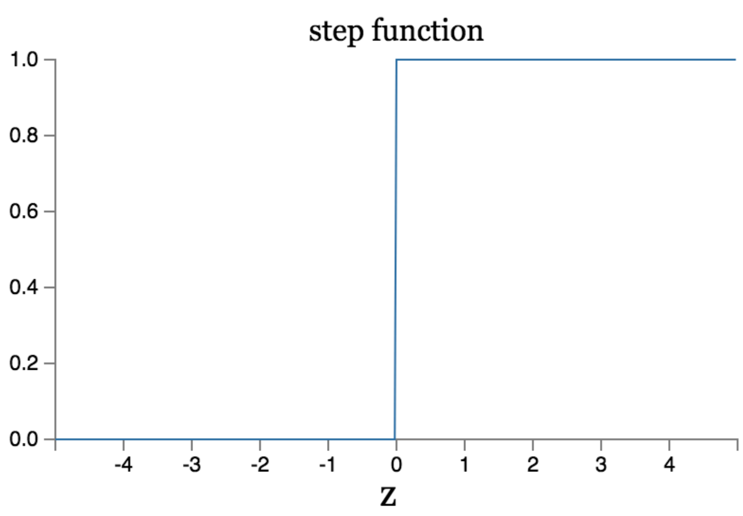
Weakness: non-smoothness

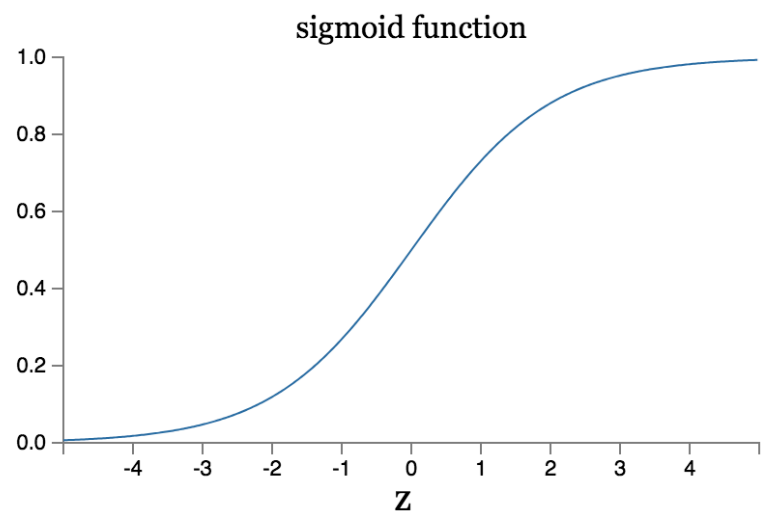
- weights or bias of any single perceptron in the network can sometimes cause the output of that perceptron to completely flip, say from 0 to 1. That flip may then cause the behaviour of the rest of the network to completely change.

Why We Need to Care this Weakness ?

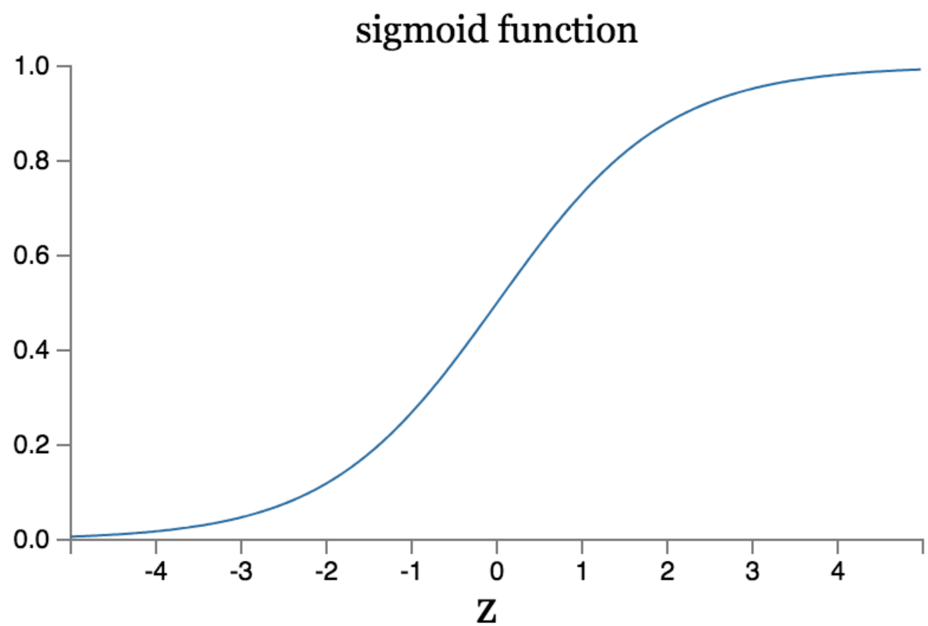
This (Non-smoothness) makes it difficult to see how to gradually modify the weights and biases so that the network gets closer to the desired behaviour.

We can overcome this problem by introducing a new type of smooth neuron called a *sigmoid* neuron.



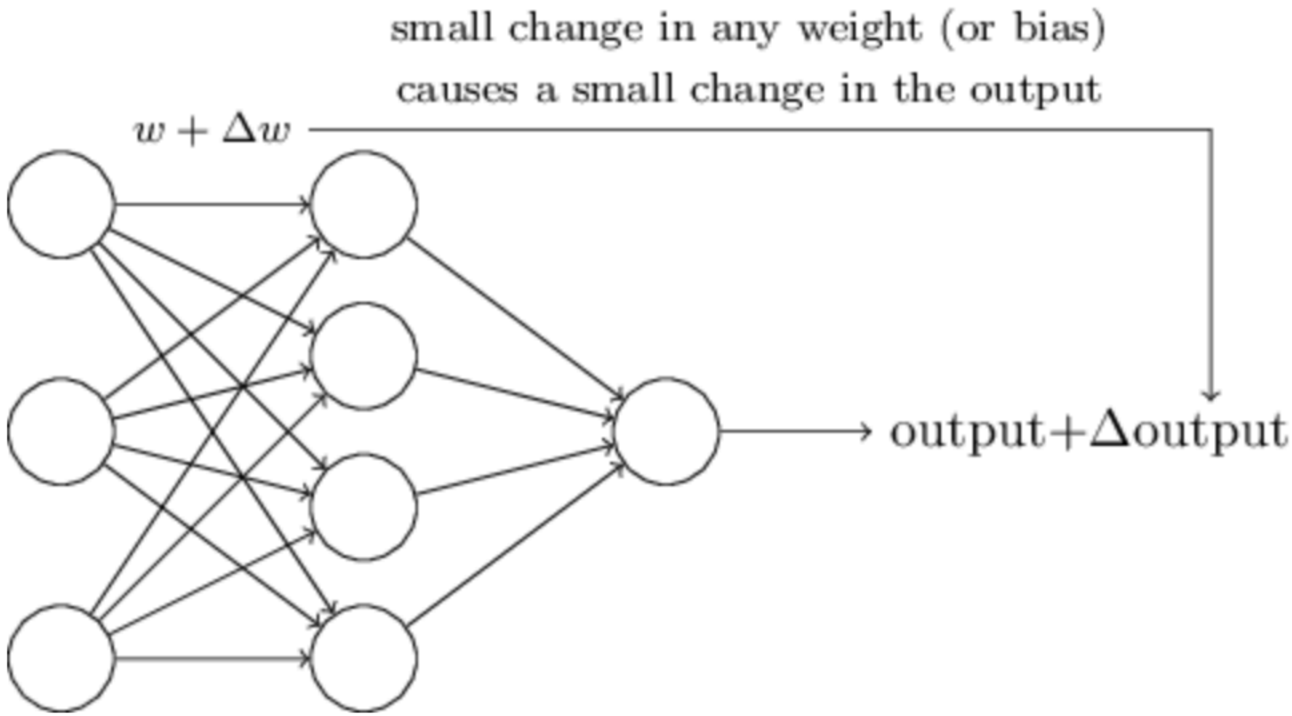


Sigmoid Neuron



$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

We can Finally Let a Small Change in Input Cause Also a Small Change in Output

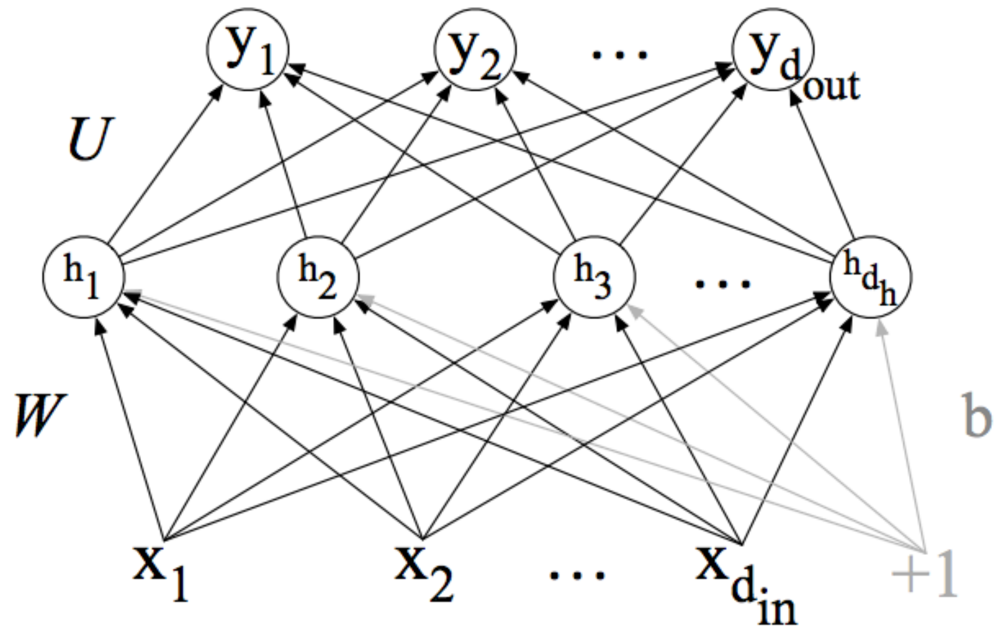


$$\Delta\text{output} \approx \sum_j \frac{\partial\, \text{output}}{\partial w_j} \Delta w_j + \frac{\partial\, \text{output}}{\partial b} \Delta b$$

Outline

- Single Neuron
- Multiple Neurons (Neural Network)
- Neural Language Model
- Summary

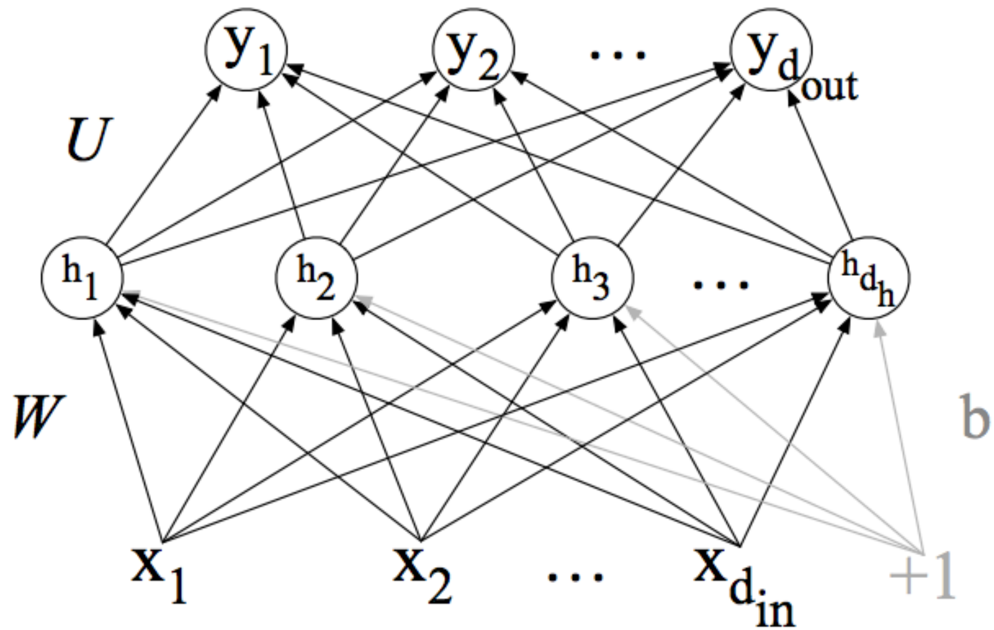
Multiple Neurons (Neural Network)



$$h = \sigma(Wx+b)$$
$$z = Uh$$
$$y = \text{softmax}(z)$$

However, z can’t be the output of the classifier, since it’s a vector of real-valued numbers, while what we need for classification is a vector of probabilities.

Softmax function



$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq D \tag{8.10}$$

Thus for example given a vector z=[0.6 1.1 -1.5 1.2 3.2 -1.1], softmax(z) is [0.055 0.090 0.0067 0.10 0.74 0.010].

Training Neural Networks

- Loss function:

$$L(\hat{y}, y) \;=\; \text{How much } \hat{y} \text{ differs from the true } y$$

- Straightforward way:

$$L_{\text{MSE}}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{m} (\hat{y}^{(m)} - y^{(i)})^2$$

- In practice, people usually will not use this MSE loss function. Instead, they use the cross-entropy loss function because it is faster to get the optimal.

Outline

- Single Neuron
- Multiple Neurons (Neural Network)
- Neural Language Model
- Summary

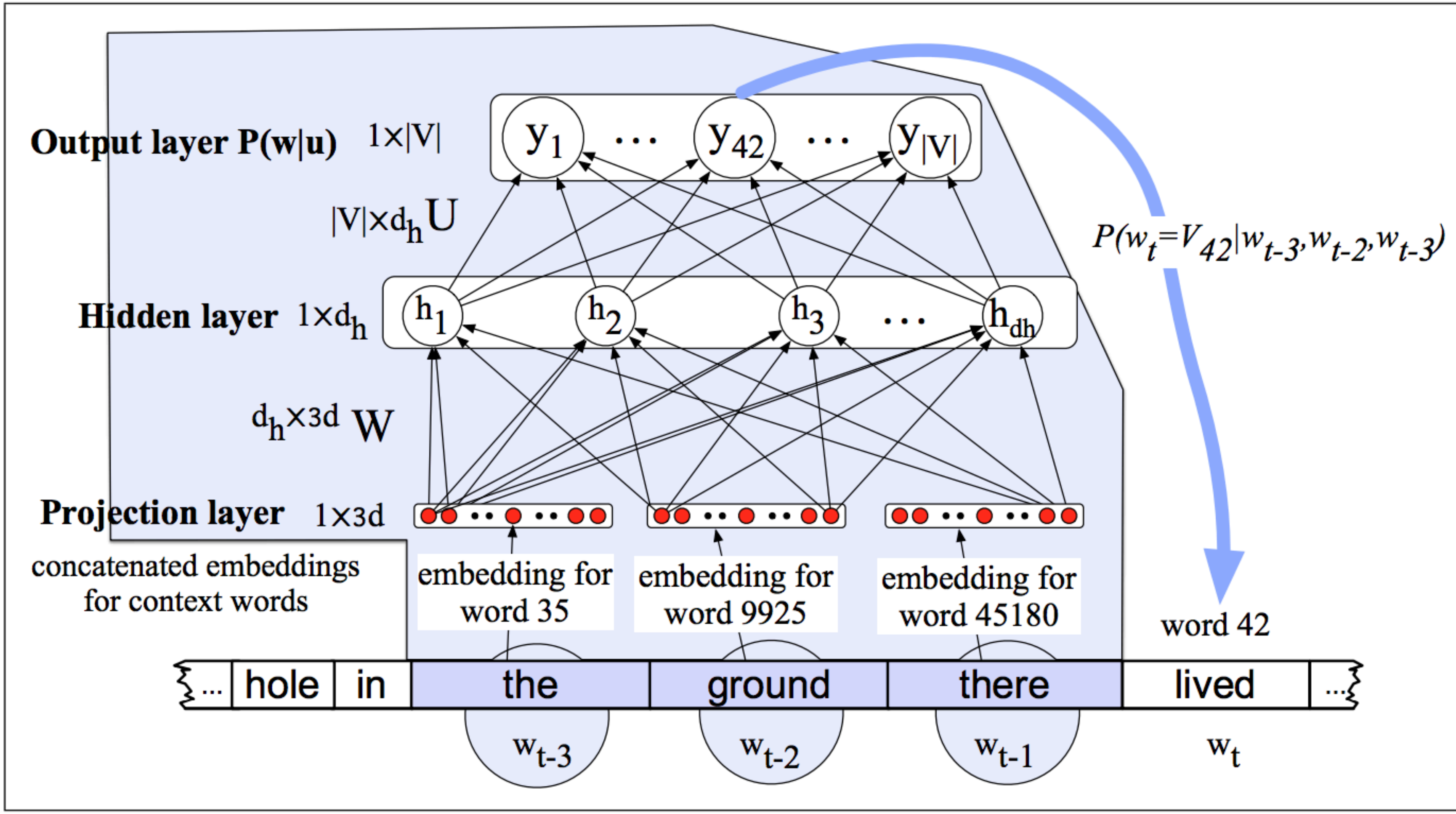
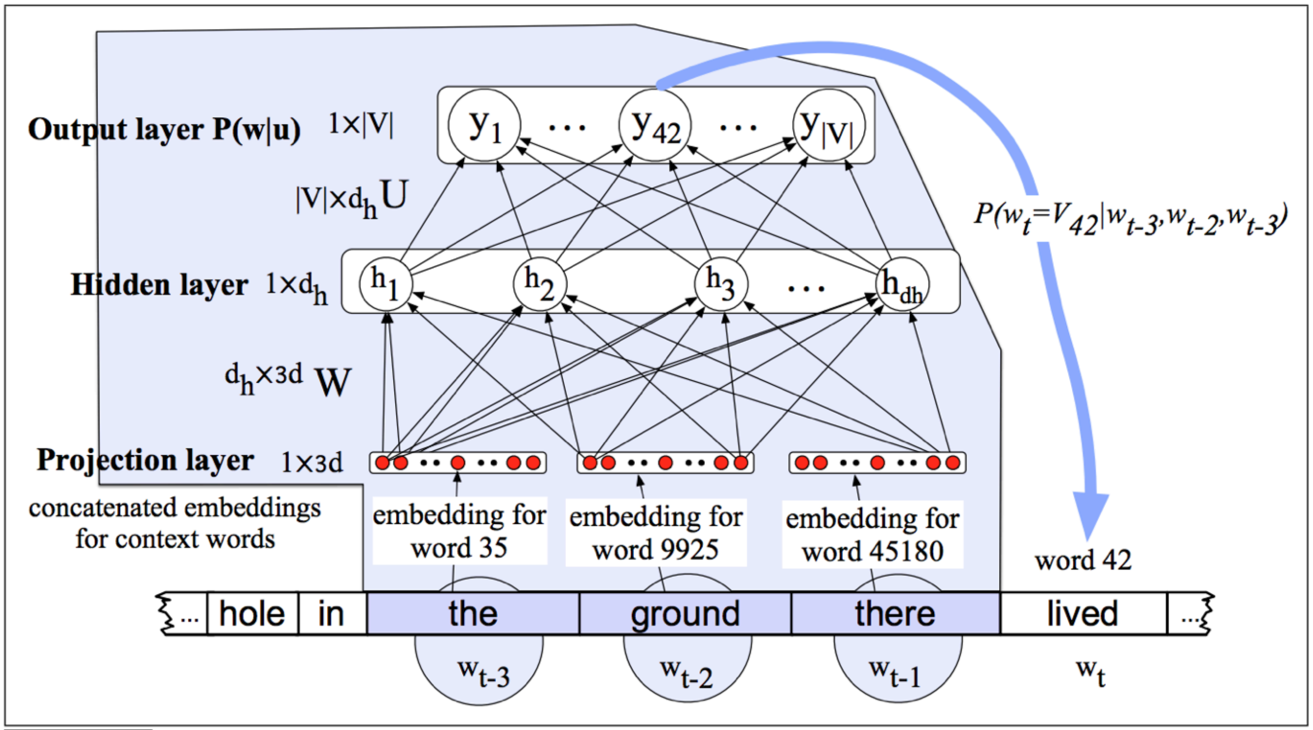


Figure 8.12 A simplified view of a feedforward neural language model moving through a text. At each timestep t the network takes the 3 context words, converts each to a d-dimensional embeddings, and concatenates the 3 embeddings together to get the $1 \times Nd$ unit input layer x for the network. These units are multiplied by a weight matrix W and bias vector b and then an activation function to produce a hidden layer h, which is then multiplied by another weight matrix U. (For graphic simplicity we don't show b in this and future pictures). Finally, a softmax output layer predicts at each node i the probability that the next word w_t will be vocabulary word V_i. (This picture is simplified because it assumes we just look up in a dictionary table E the "embedding vector", a d-dimensional vector representing each word, but doesn't yet show us how these embeddings are learned.)

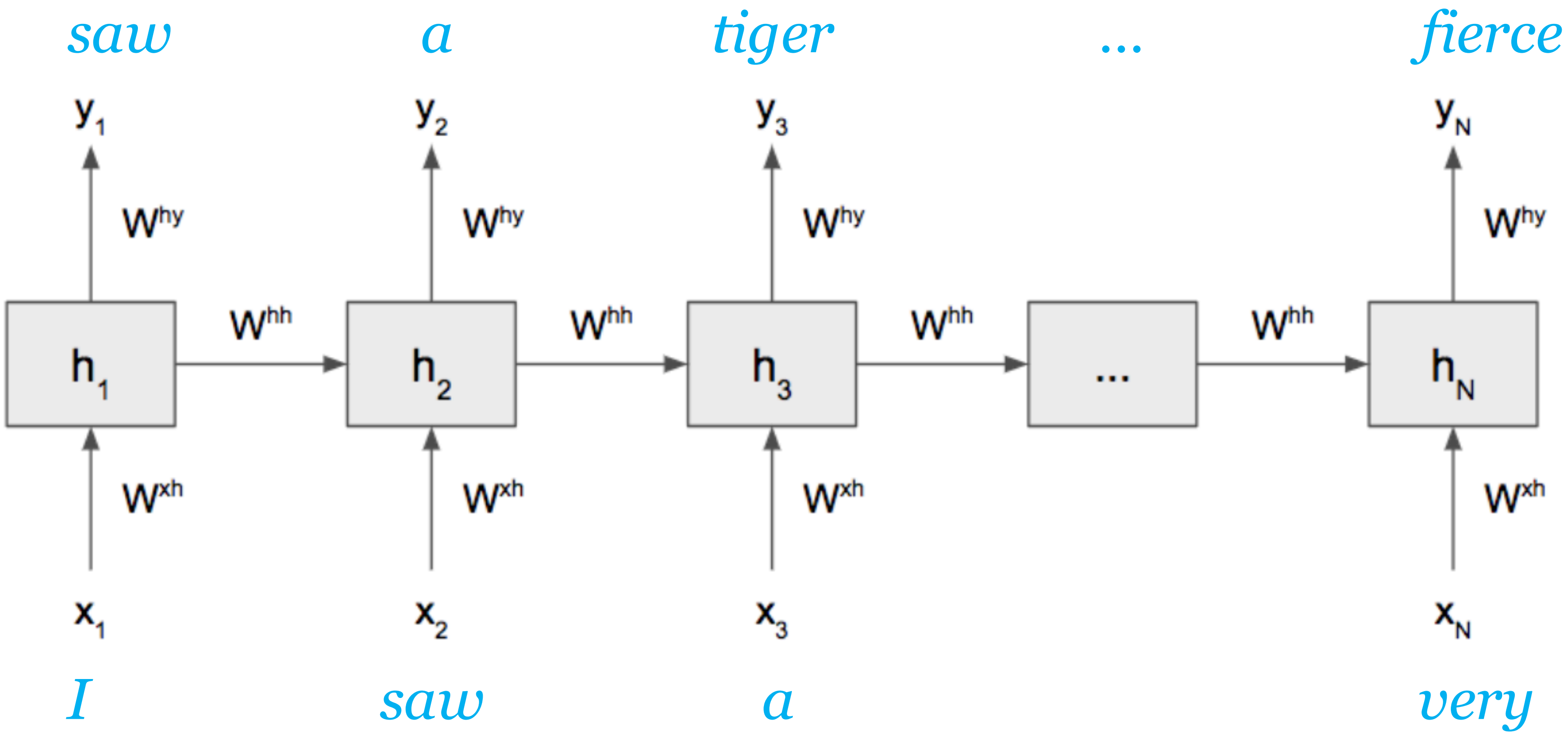
Limitation of Naïve Neural language model – limited context information



In practise, people usually use recurrent neural language model that can take the long dependency into account. For example:

I saw a tiger which was really very *talkative or fierce* .

Recurrent Neural Language Model



Outline

- Single Neuron
- Multiple Neurons (Neural Network)
- Neural Language Model
- Summary

Summary

- Neural networks are built out of neural units, originally inspired by human neurons but now simple an abstract computational device.
- Each neural unit multiplies input values by a weight vector, adds a bias, and then applies a non-linear activation function like sigmoid, tanh, or rectified linear.
- Neural language modeling uses a network as a probabilistic classifier, to compute the probability of the next word given the previous N word.
- Neural language models make use of embeddings, dense vectors of between 50 and 500 dimensions that represent words in the input vocabulary.

Reference

1. http://neuralnetworksanddeeplearning.com/