NN_NLP CHAPTER 4

Feed Forward Neural Networks

马新宇、刘嘉铭

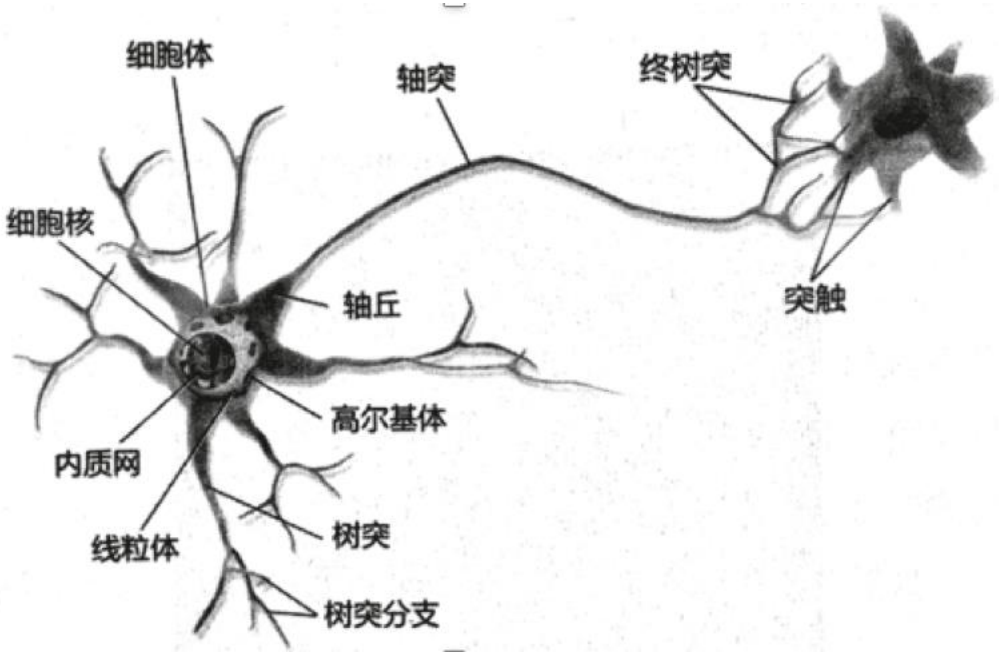
2018/9/17

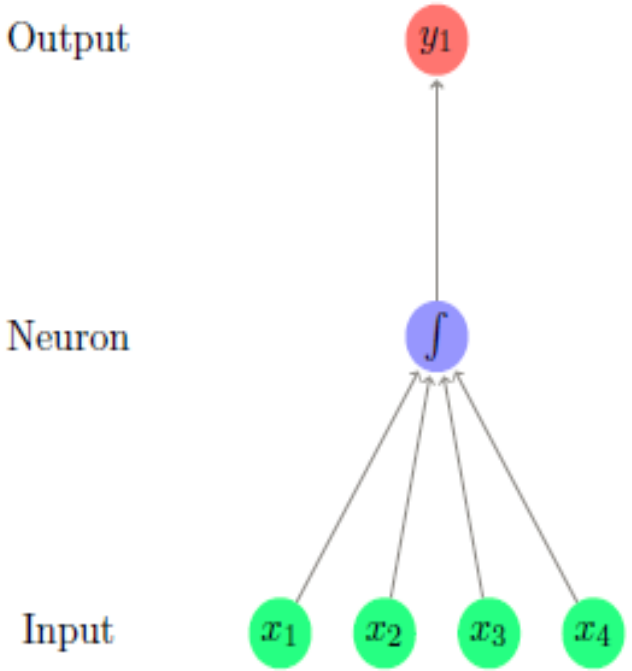
Outline

- 神经网络的灵感来源（chapter4.1）
- 前馈神经网络基本介绍（chapter4.1）
- 其数学表达形式（chapter4.2）
- 神经网络的表达能力（chapter4.3）
- 神经网络的激活函数（chapter4.4）
- 神经网络的损失函数（chapter4.5）
- 神经网络的正则化（chapter4.6）
- 利用神经网络计算相似度（chapter4.7）
- 神经网络的嵌入层（chapter4.8）

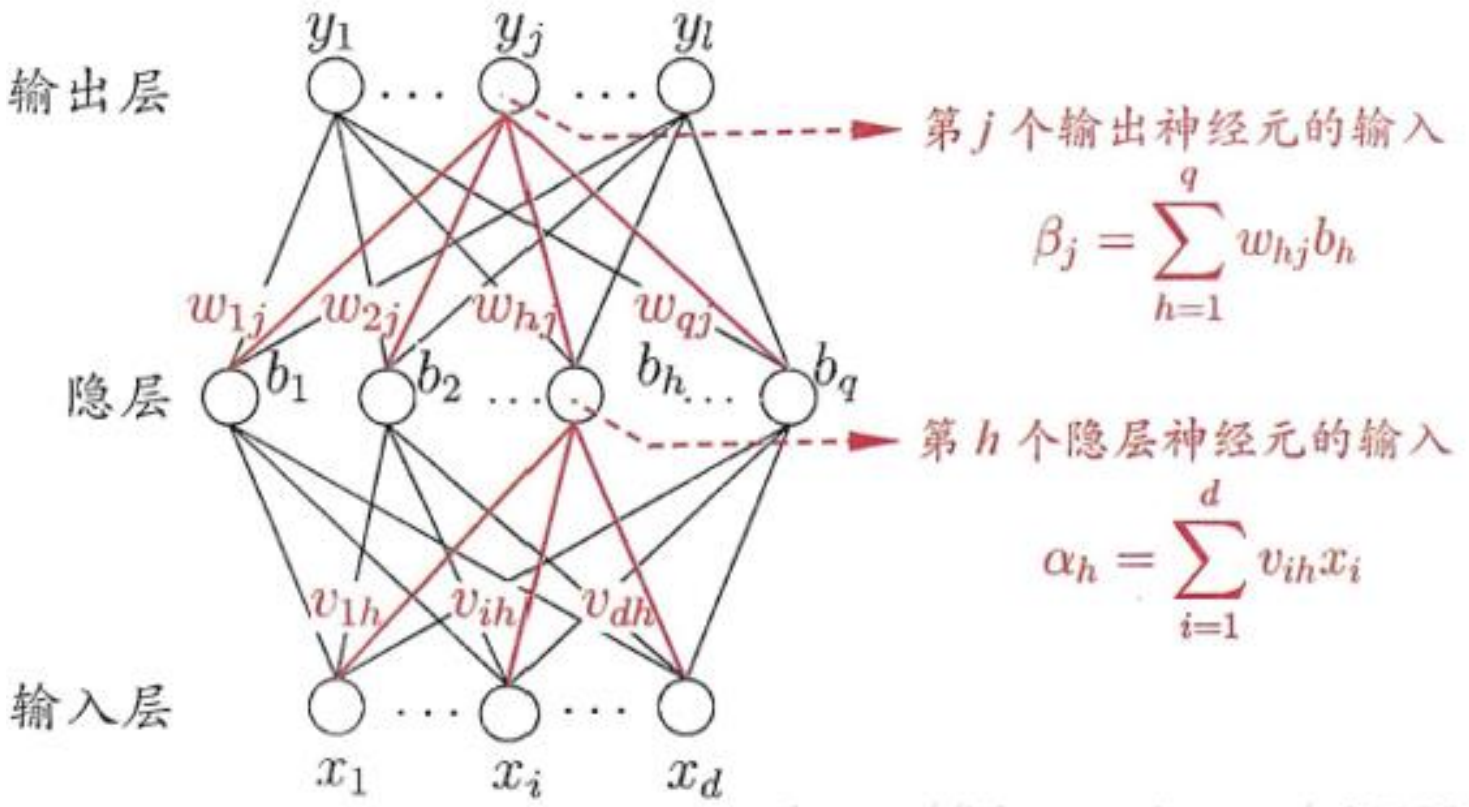
4.1 A Brain-inspired Metaphor

- 生物神经元 vs 神经网络中的神经元





4.2 Feed forward neural network



4.2 In mathematical notation

$$NN_{MLP1}(x) = g(xW^1 + b^1)W^2 + b^2$$

$$s.t \quad x \in R^{d_{in}}, W^1 \in R^{d_{in}\mathrm{x}d_1}, b^1 \in R^{d_1},$$
$$W^2 \in R^{d_1\mathrm{x}d_2}, b^2 \in R^{d_2}$$

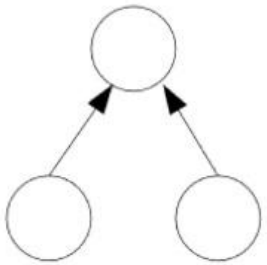
- 权重矩阵W和偏置项b定义了线性变换
- g定义了非线性激活函数
- 当输出是一维时，标量可用于回归问题
- 当输出是符号或二值时，可用于二分类问题
- 当输出是概率分布时，可用于多分类

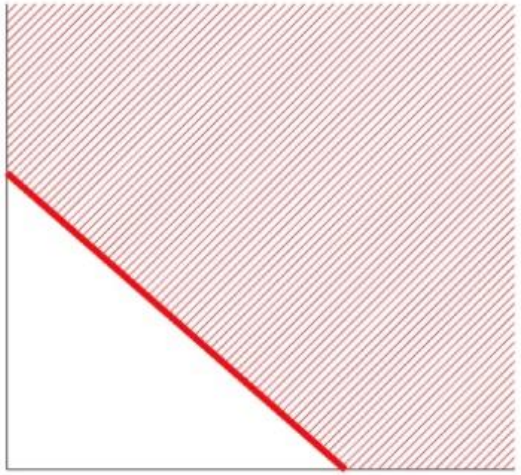
4.3 Representation power

- Hornik[1989]等和Cybenko[1989]等给出了证明：
  - 拥有至少一个隐层的神经网络是一个通用的近似器(universal approximator)，它可以近似任何连续函数。$|f(x) - g(x)| < \varepsilon$
- 为什么多层?
  - 多数情况多层神经网络表达出的函数更加平滑，学习起来更加容易
- 实际中存在一些复杂神经网络无法用层数更少的网络近似，除非层的大小是指数级的。

4.3 Representation power

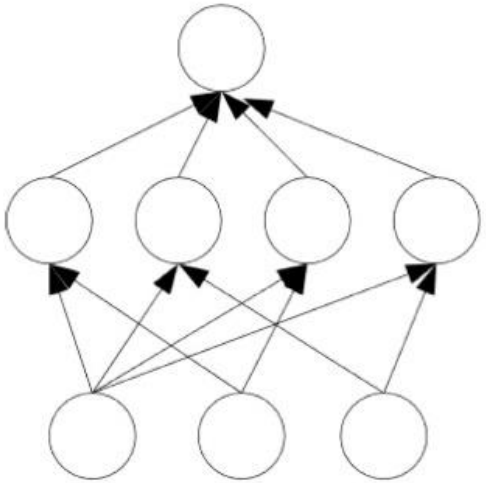
1 layer of trainable weights

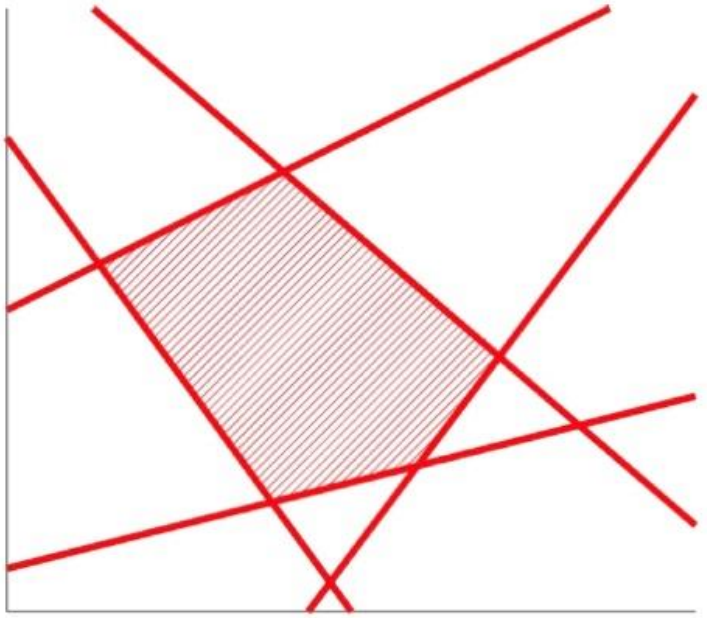




separating hyperplane

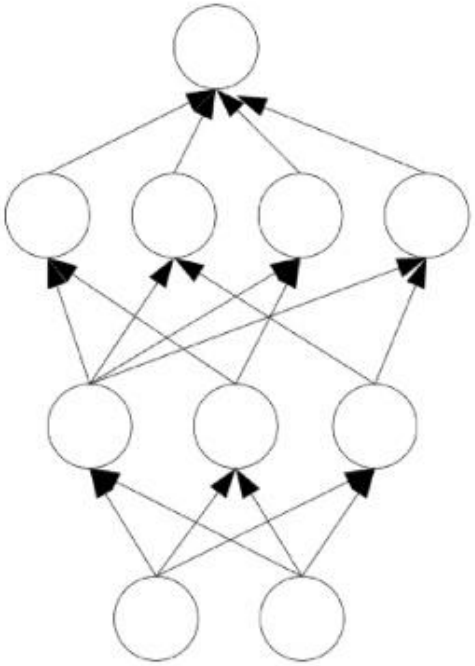
4.3 Representation power

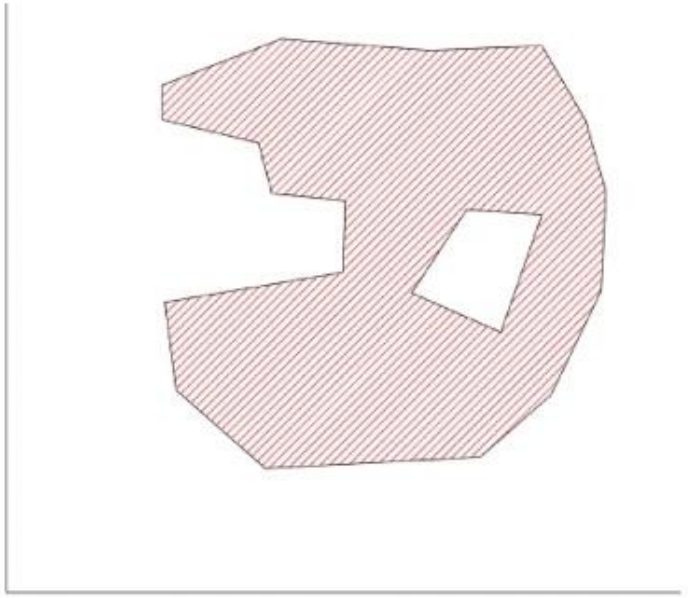




convex polygon region

4.3 Representation power

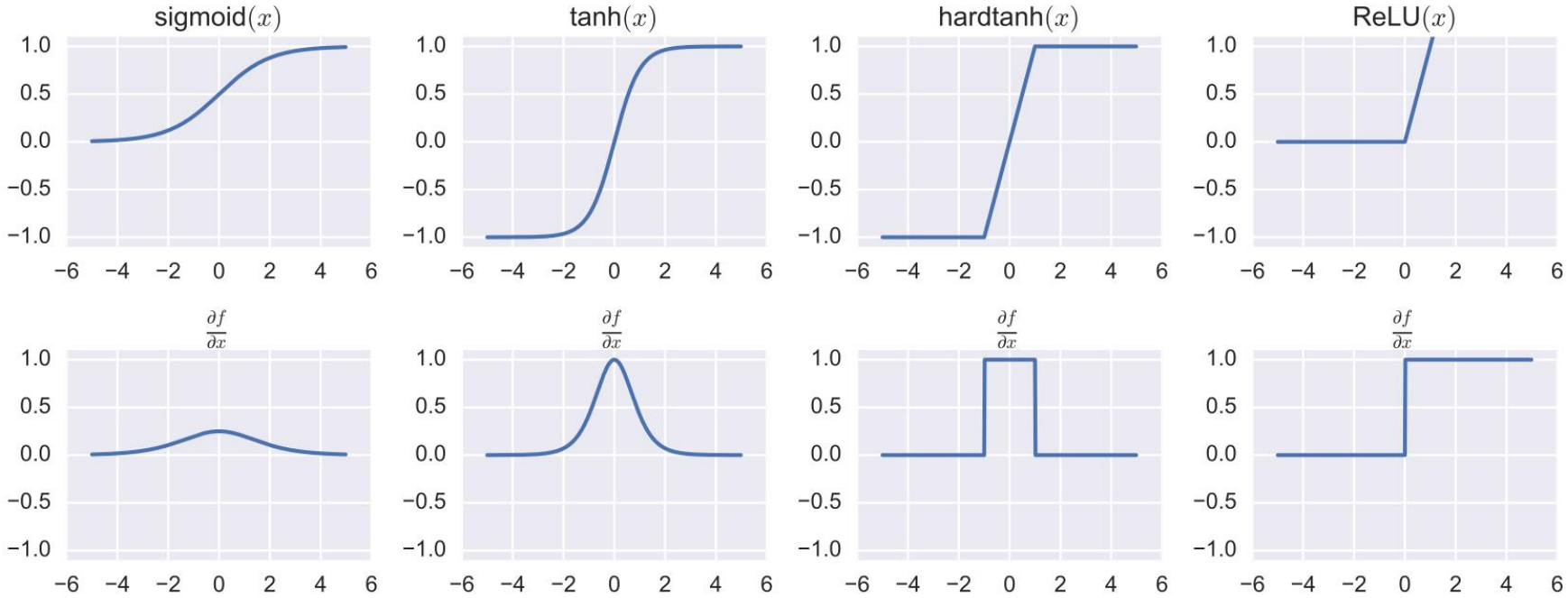




composition of polygons:
convex regions

4.4 Common Nonlinearities

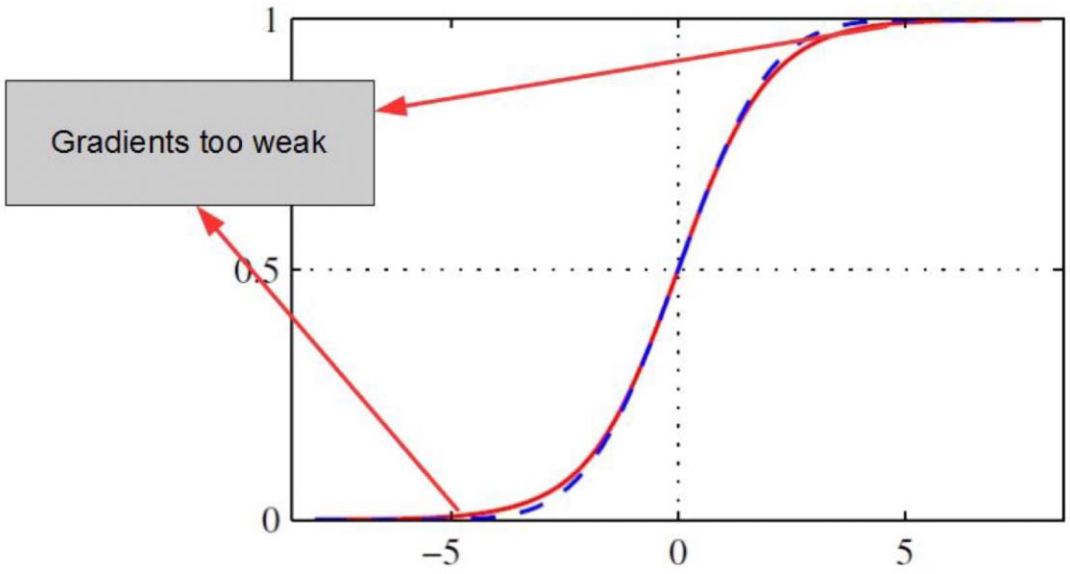
The non-linearity g can take many forms. There is currently no good theory as to which non-linearity to apply in which conditions, and choosing the correct non-linearity for a given task is for the most part an empirical question.



4.4 Common Nonlinearities

Sigmoid

$$\sigma(x) = 1/(1 + e^{-x})$$



- Problem: Saturate across most of their domain, strongly sensitive only when z is closer to zero
- Sigmoids saturate and kill gradients.
- Sigmoid outputs are not zero-centered.
- Not Recommend!!

4.4 Common Nonlinearities

About Zero-Centered

- *Sigmoid outputs are not zero-centered*. This is undesirable since neurons in later layers of processing in a Neural Network (more on this soon) would be receiving data that is not zero-centered. This has implications on the dynamics during gradient descent, because if the data coming into a neuron is always positive (e.g. $x > 0$ elementwise in $f = w^T x + b$)), then the gradient on the weights w will during backpropagation become either all be positive, or all negative (depending on the gradient of the whole expression f). This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights. However, notice that once these gradients are added up across a batch of data the final update for the weights can have variable signs, somewhat mitigating this issue. Therefore, this is an inconvenience but it has less severe consequences compared to the saturated activation problem above.

Why would having all $x > 0$ (elementwise) lead to all-positive or all-negative gradients on w?

4.4 Common Nonlinearities

About Zero-Centered

Sigmoid outputs are not zero-centered. Sigmoid 的输出不是0均值的，这是我们不希望的，因为这会导致后层的神经元的输入是非0均值的信号，这会对梯度产生影响：假设后层神经元的输入都为正(e.g. x>0 elementwise in $f = w_T x + b$),那么对w求局部梯度则都为正，这样在反向传播的过程中w要么都往正方向更新，要么都往负方向更新，导致有一种捆绑的效果，使得收敛缓慢。

当然了，如果你是按batch去训练，那么每个batch可能得到不同的符号（正或负），那么相加一下这个问题还是可以缓解。因此，非0均值这个问题虽然会产生一些不好的影响，不过跟上面提到的 kill gradients 问题相比还是要好很多的。

$$f = \sum w_i x_i + b$$

$$\frac{df}{dw_i} = x_i$$

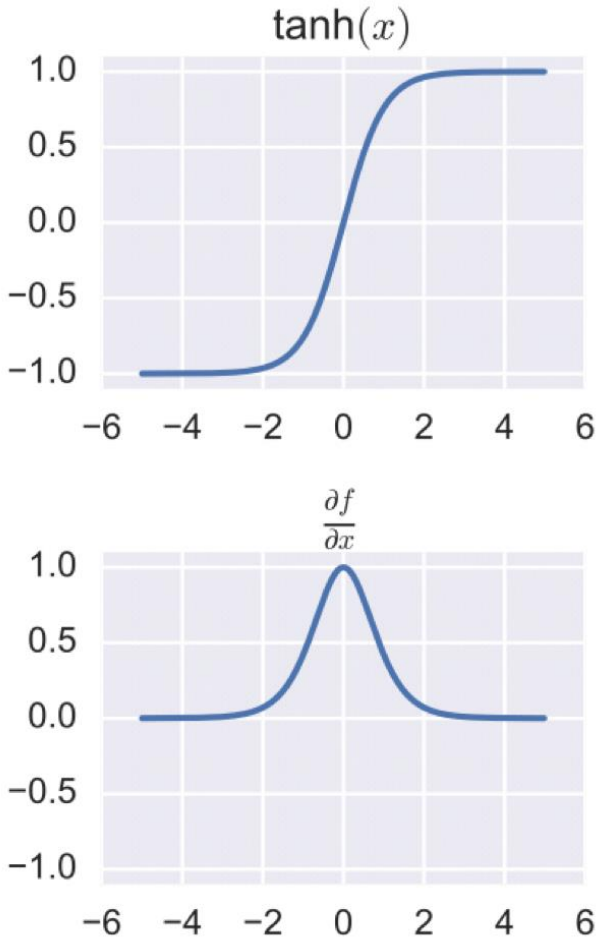
$$\frac{dL}{dw_i} = \frac{dL}{df}\frac{df}{dw_i} = \frac{dL}{df}x_i$$

because $x_i > 0$, the gradient $\frac{dL}{dw_i}$ always has the same sign as $\frac{dL}{df}$ (all positive or all negative).

4.4 Common Nonlinearities

Tanh $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$

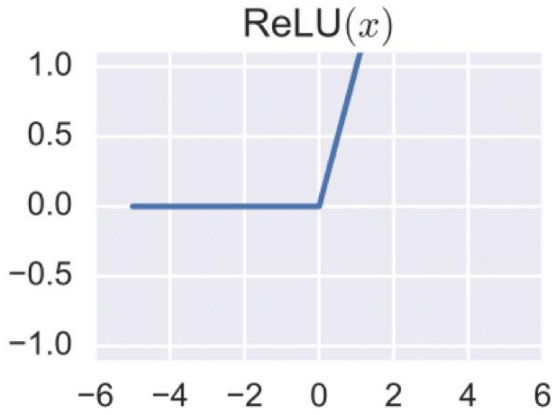
- Related to sigmoid: tanh(x)=$2\sigma(2x) - 1$
- Positives: Squashes out to range [-1, 1], outputs are zero-centered
- Negative: Also satuates
- Better than sigmoid, when activations are small



4.4 Common Nonlinearities

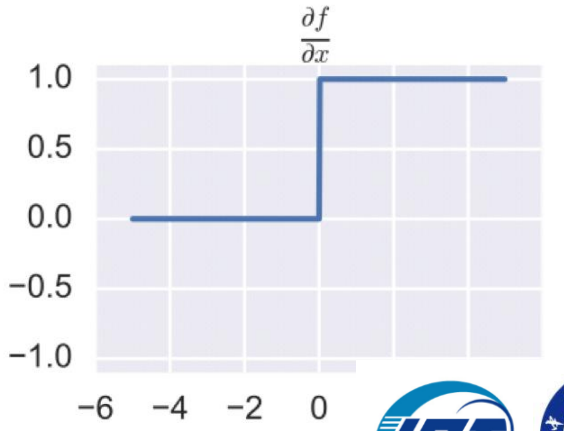
ReLU

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0 & x < 0 \\ x & \text{otherwise} \end{cases}$$



- Positives:
- Give large and consistent gradients(does not saturate) when active
- Efficient to optimize, converges much faster than sigmoid or tanh

- Negatives:
- Non zero centered output
- Units “die”: when inactive they will never update



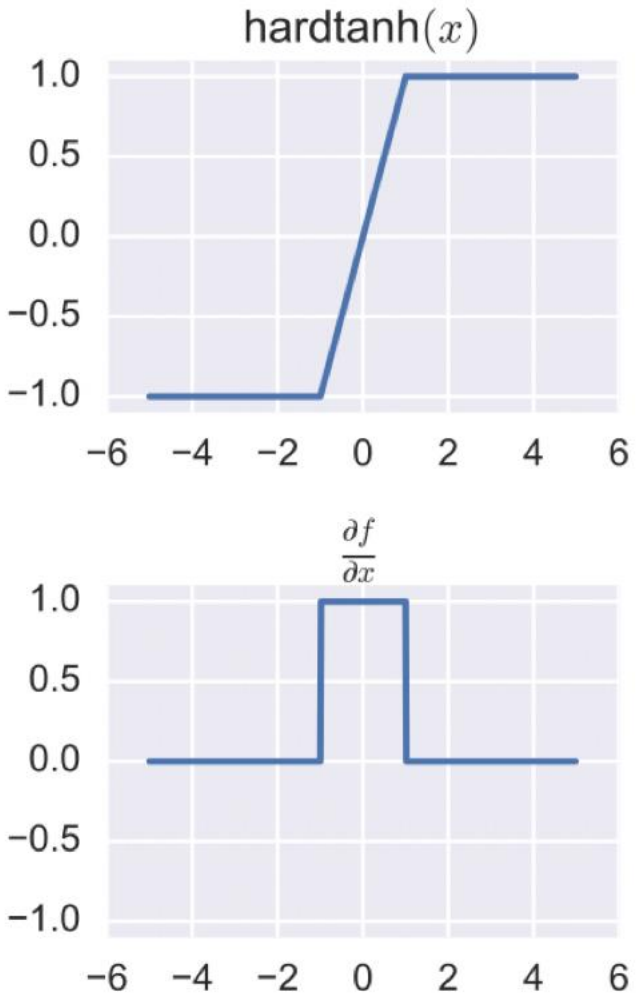
- Good Practice:
- Initialize b to a small positive value
- Ensures units are initially active for most inputs and derivations can pass through

4.4 Common Nonlinearities

Hard tanh

The hard-tanh activation function is an approximation of the tanh function which is faster to compute the derivatives.

$$\text{hardtanh}(x) = \begin{cases} -1 & x < -1 \\ 1 & x > 1 \\ x & \text{otherwise} \end{cases}$$



4.5 Loss Functions

Mean squared error： $L(\tilde{y}, y) = \sum_i (y_{[i]} - \widetilde{y_{[i]}})^2$

Binary cross entropy： $L(\tilde{y}, y) = -ylog\tilde{y} - (1 - \mathrm{y})\log(1 - \tilde{y})$

Categorical cross-entropy： $L(\tilde{y}, y) = -\sum_i y_{[i]} \log(\widetilde{y_{[i]}})$

4.6 Regularization And Dropout

L1： $R_{L1}(W) = \left||W|\right|_1^2 = \sum_{i,j} |W_{[i,j]}|$

L2： $R_{L2}(W) = \left||W|\right|_2^2 = \sum_{i,j} (W_{[i,j]})^2$

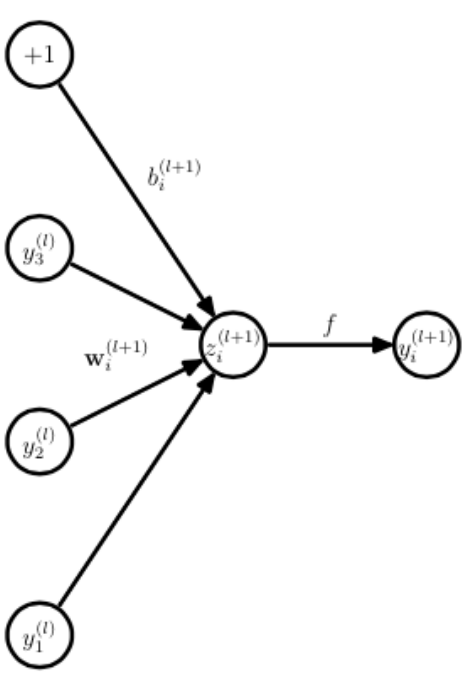
In particular, L2 regularization, also called weight decay is effective for achieving good generalization performance in many cases, and tuning the regularization strength is advisable.

Dropout：randomly dropping (setting to 0) half of the neurons in the network in each training example.

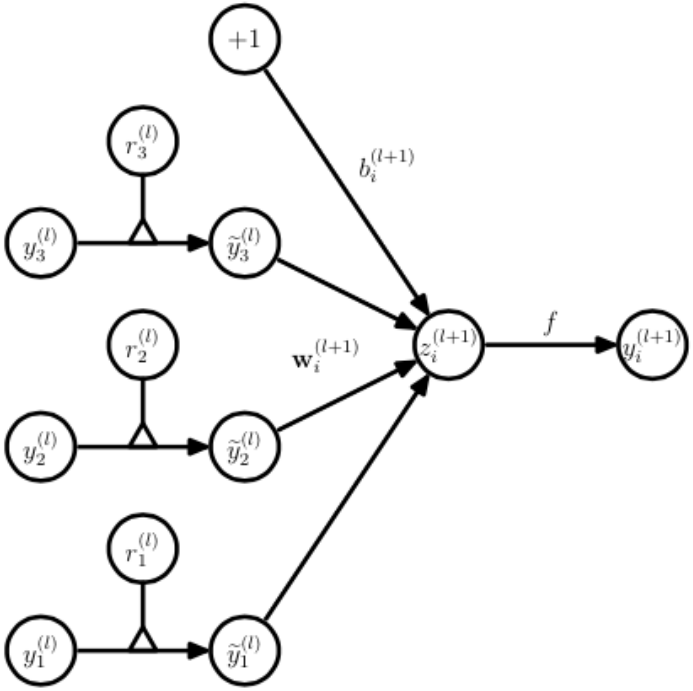
在训练时，每个神经单元以概率p被保留(dropout丢弃率为1-p)；在测试阶段，每个神经单元都是存在的，权重参数w要乘以p，成为：pw。

4.6 Regularization And Dropout

Dropout



(a) Standard network



(b) Dropout network

4.6 Regularization And Dropout

Dropout

$$
\begin{aligned}
\mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) &= \mathbf{y} \\
\mathbf{h^1} &= g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1}) \\
\mathbf{h^2} &= g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2}) \\
\mathbf{y} &= \mathbf{h^2}\mathbf{W^3}
\end{aligned}
$$

Standard network

$$
\begin{aligned}
\mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) &= \mathbf{y} \\
\mathbf{h^1} &= g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1}) \\
\mathbf{m^1} &\sim \mathrm{Bernouli}(r^1) \\
\tilde{\mathbf{h}}^{\mathbf{1}} &= \mathbf{m^1} \odot \mathbf{h^1} \\
\mathbf{h^2} &= g^2(\tilde{\mathbf{h}}^{\mathbf{1}}\mathbf{W^2} + \mathbf{b^2}) \\
\mathbf{m^2} &\sim \mathrm{Bernouli}(r^2) \\
\tilde{\mathbf{h}}^{\mathbf{2}} &= \mathbf{m^2} \odot \mathbf{h^2} \\
\mathbf{y} &= \tilde{\mathbf{h}}^{\mathbf{2}}\mathbf{W^3}
\end{aligned}
$$

Dropout network

4.7 Similarity And Distance Layers

Background:
We sometimes wish to calculate a scalar based on two vectors, such that the value reflect similarity, compatibility, distance.

Fixed Functions

Dot product:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \boldsymbol{u} \cdot \boldsymbol{v}$$

Euclidean Distance:

$$\text{dist}(\mathbf{u}, \mathbf{v}) = \sqrt{(\boldsymbol{u} - \boldsymbol{v}) \cdot (\boldsymbol{u} - \boldsymbol{v})}$$

4.7 Similarity And Distance Layers

Trainable Forms

Train a matrix: $\boldsymbol{M} \in R^{d \times d}$

Similarity:

$$\text{sim}(\mathbf{u}, \mathbf{v}) = \boldsymbol{u}\boldsymbol{M}\boldsymbol{v}^T$$

Distance:

$$\text{dist}(\mathbf{u}, \mathbf{v}) = (\boldsymbol{u} - \boldsymbol{v})\boldsymbol{M}(\boldsymbol{u} - \boldsymbol{v})^T$$

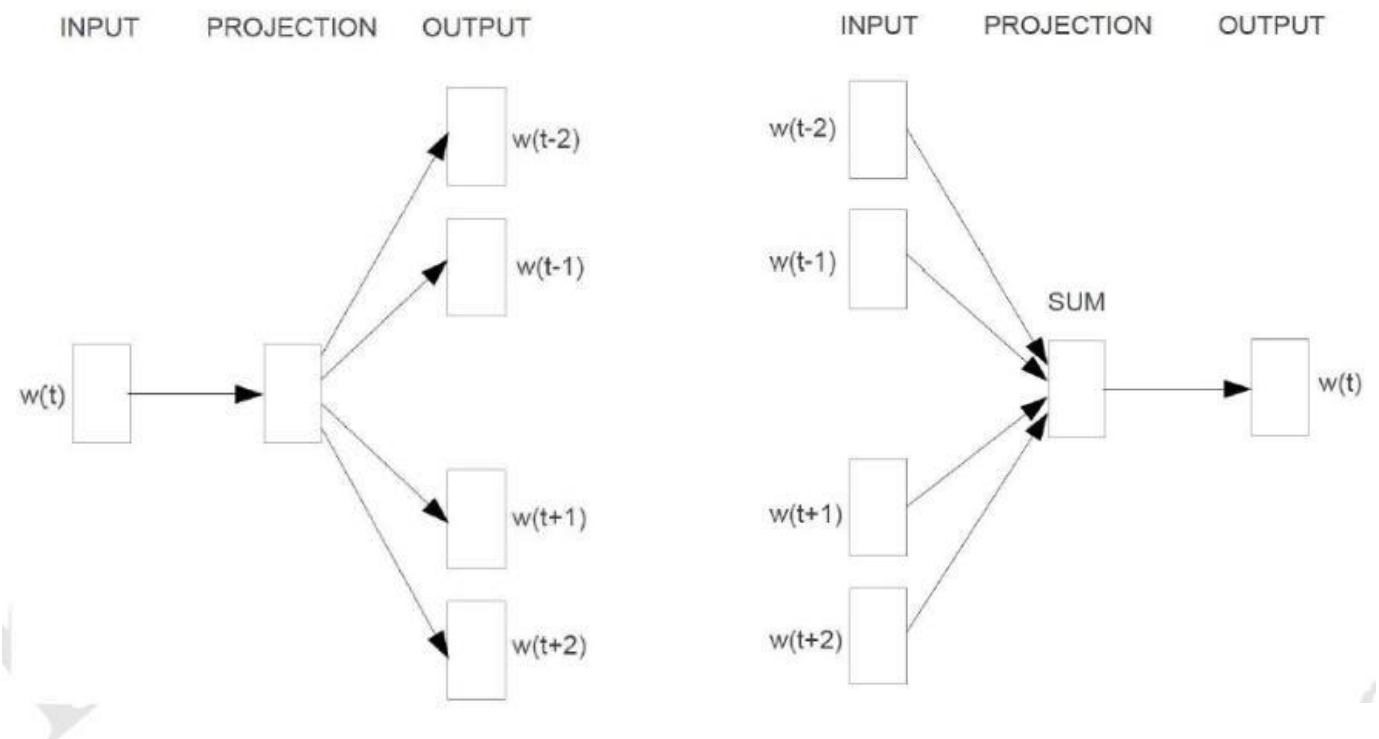
Use MLP:

Input: concatenation of $\boldsymbol{u}$ and $\boldsymbol{v}$

Output: a scalar(distance or similarity)

4.8 Embedding Layers

The mapping from a symbolic feature values such as “word number 1249” to d-dimensional vectors is performed by an embedding layer.

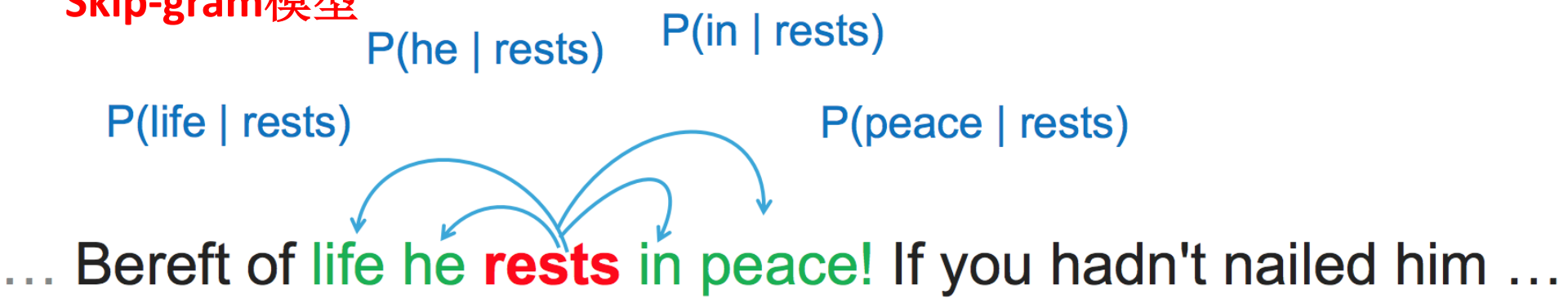


Skip-gram

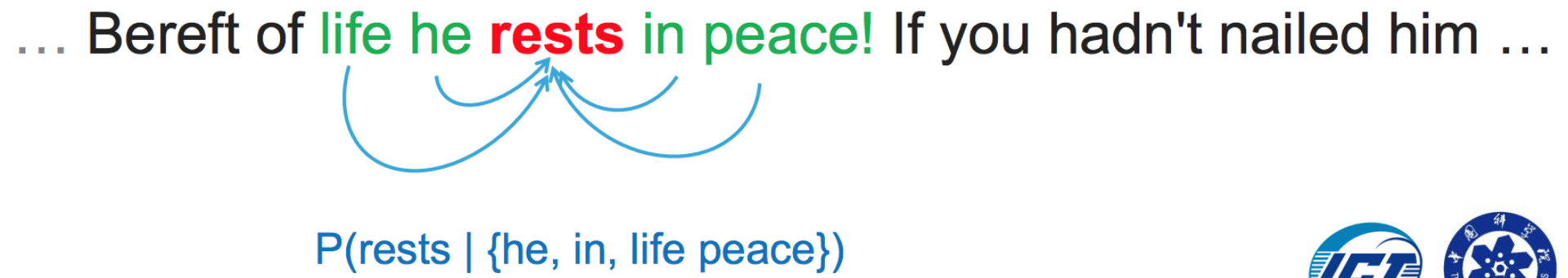
CBOW

4.8 Embedding Layers

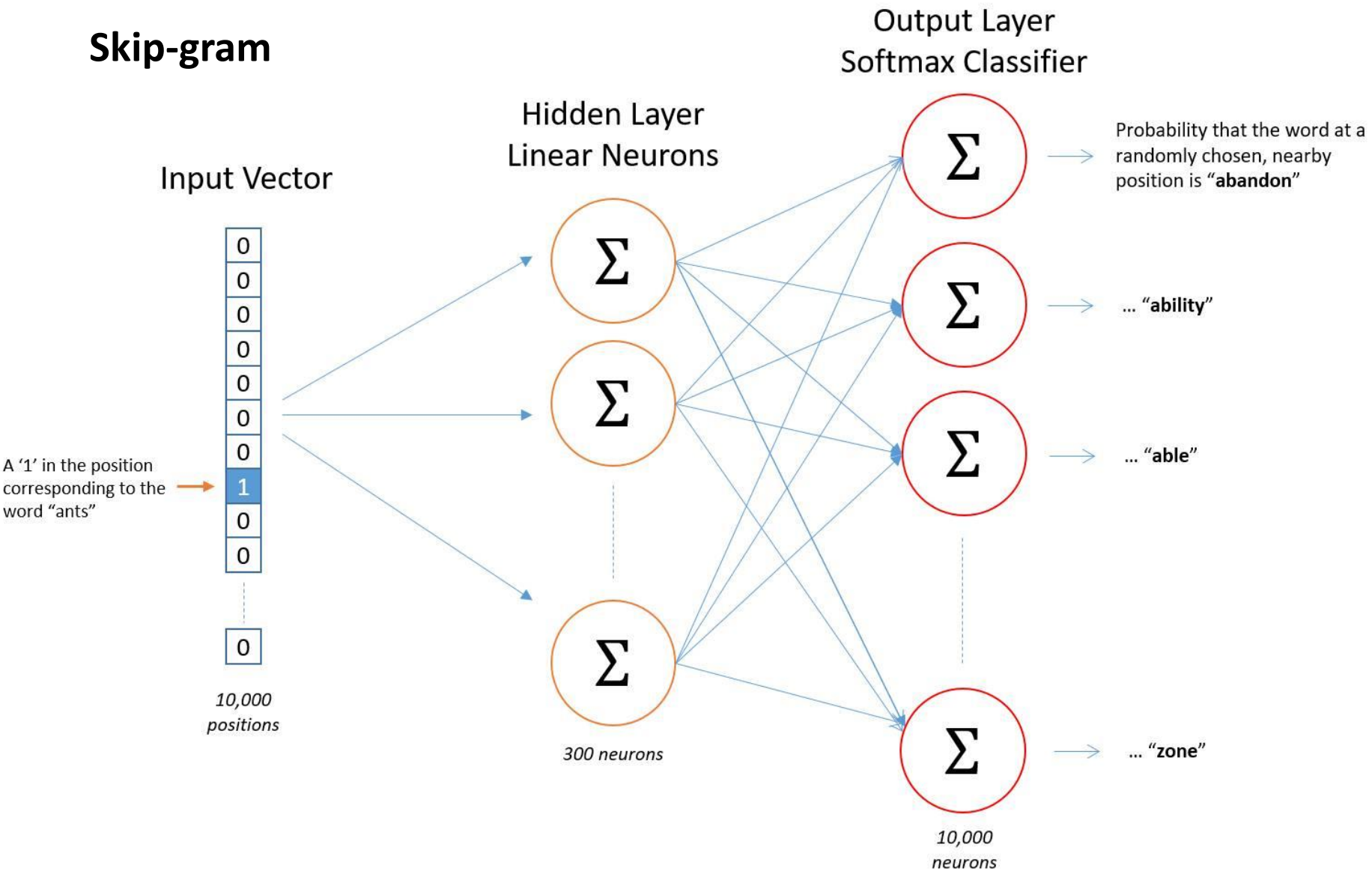
Skip-gram模型

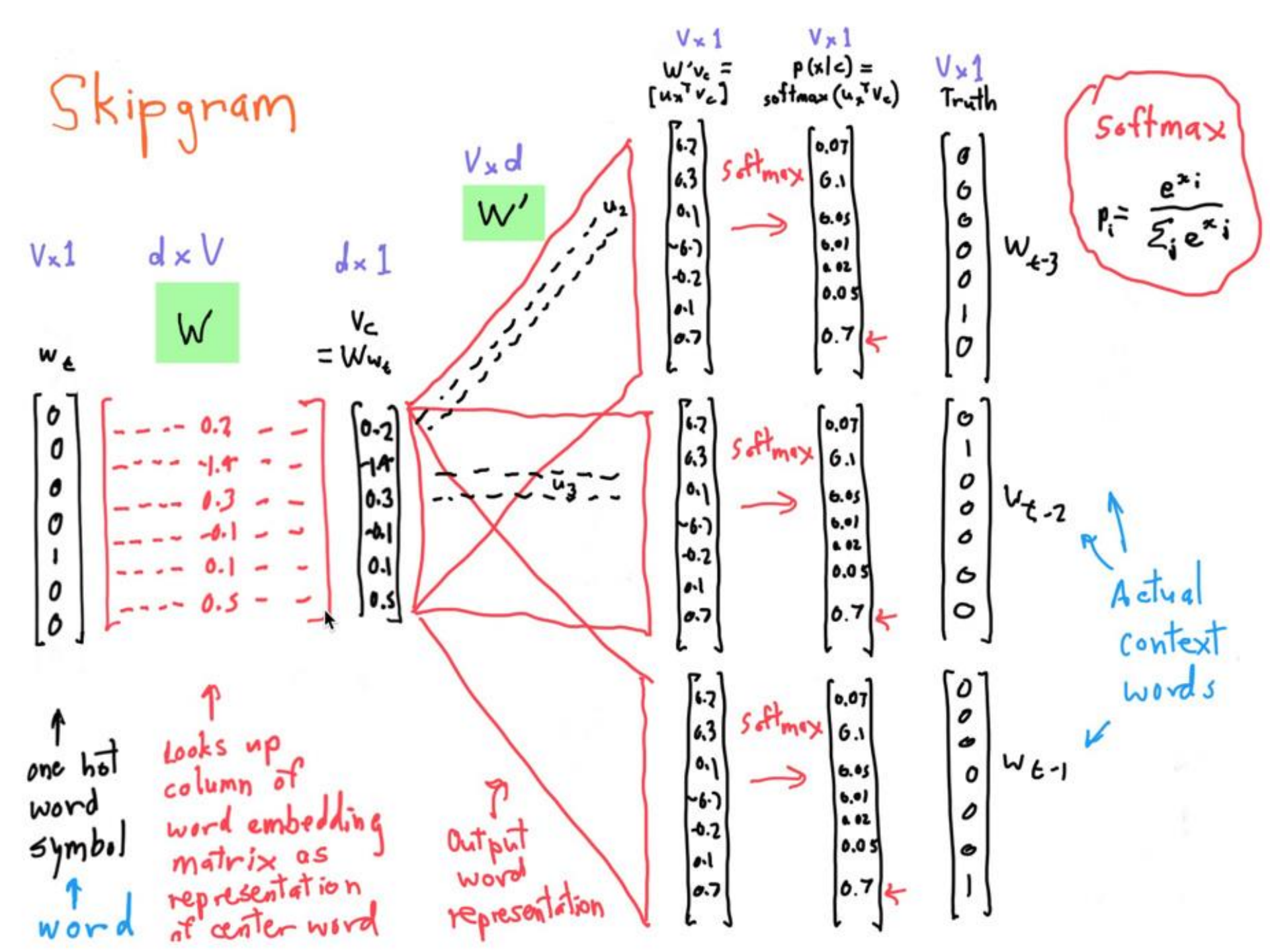


CBOW模型(Continuous bag of words)



Skip-gram


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
A '1' in the position corresponding to the word "ants"
0
0
0
0
0
0
0
1
0
0
0
10,000 positions
300 neurons
10,000 neurons


Skipgram
V×1
d×V
W
d×1
$v_c = W w_t$
w_t
V×d
W'
u_2
u_3
V×1
$W' v_c = [u_x^T v_c]$
$p(x|c) = \text{softmax}(u_x^T v_c)$
V×1
Truth
Softmax
$p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
softmax
W_{t-3}
W_{t-2}
W_{t-1}
Actual context words
one hot word symbol
word
Looks up column of word embedding matrix as representation of center word
Output word representation