Deep Learning for Natural Language Processing

Dr. Minlie Huang (黄民烈)

aihuang@tsinghua.edu.cn

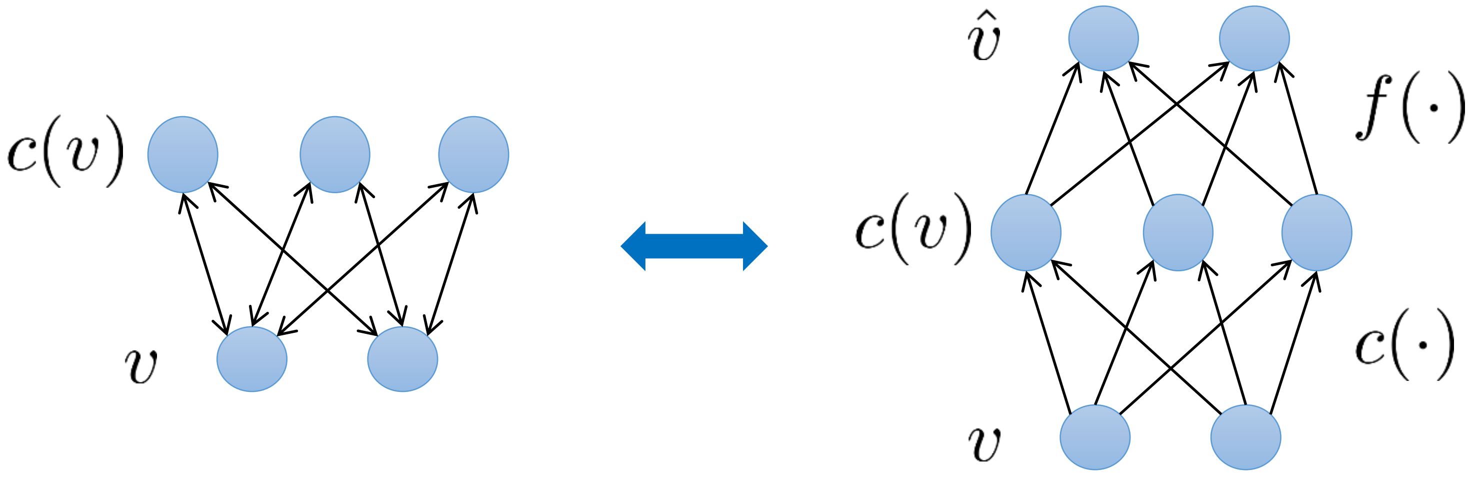
Computer Science Department

Tsinghua University

Homepage: http://coai.cs.Tsinghua.edu.cn/hml

Recursive Networks for NLP

Auto-encoder

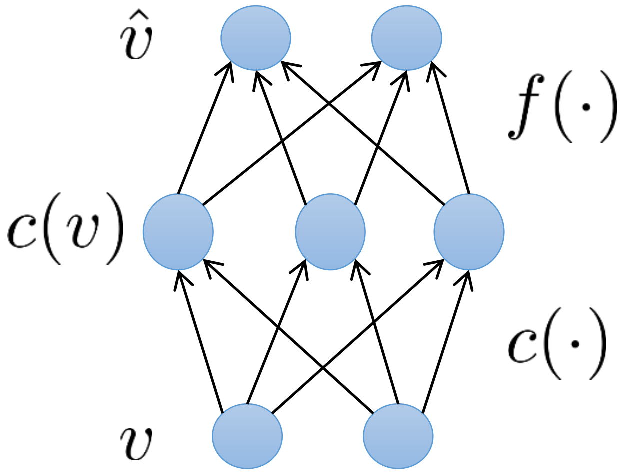


- Encode the input ***v*** into some representation ***c(v)*** so that the input can be reconstructed from that representation
  - Encoding function ***c(v)***
  - Decoding function ***f(c(v))***

Encoding and decoding functions

- Nonlinear function

$$c(v) = sigmoid(W_1 v + \theta)$$
$$f(c) = sigmoid(W_2 c + \eta)$$



- If ***c*** and ***f*** are binary, then the functions can be used as probabilities

Loss function

- Minimize the reconstruction error or the negative data log-likelihood

$$RE = -\langle \ln P(v|c(v)) \rangle$$

$\langle \cdot \rangle$: average over samples

  - Gaussian probability (v is real)

$$P(v|c(v)) \propto \exp\left(\frac{-\|v - f(c(v))\|^2}{2\sigma^2}\right)$$

then

$$RE = \langle \|v - f(c(v))\|^2 \rangle$$

  - Binomial probability (v is binary)

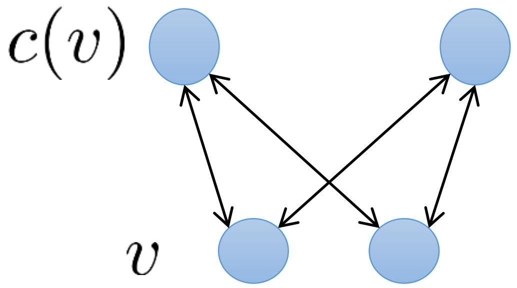
$$P(v|c(v)) \propto \Pi_i f_i(c(v))^{v_i} (1 - f_i(c(v)))^{1-v_i}$$

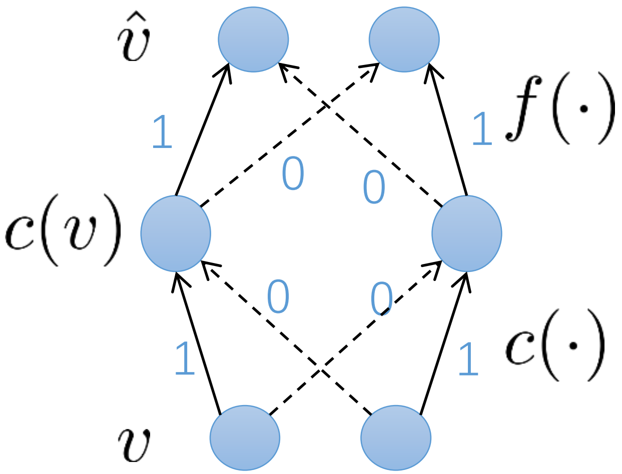
then

$$RE = -\langle \sum_i (v_i \ln f_i(c(v)) + (1 - v_i) \ln(1 - f_i(c(v)))) \rangle$$

A trivial solution

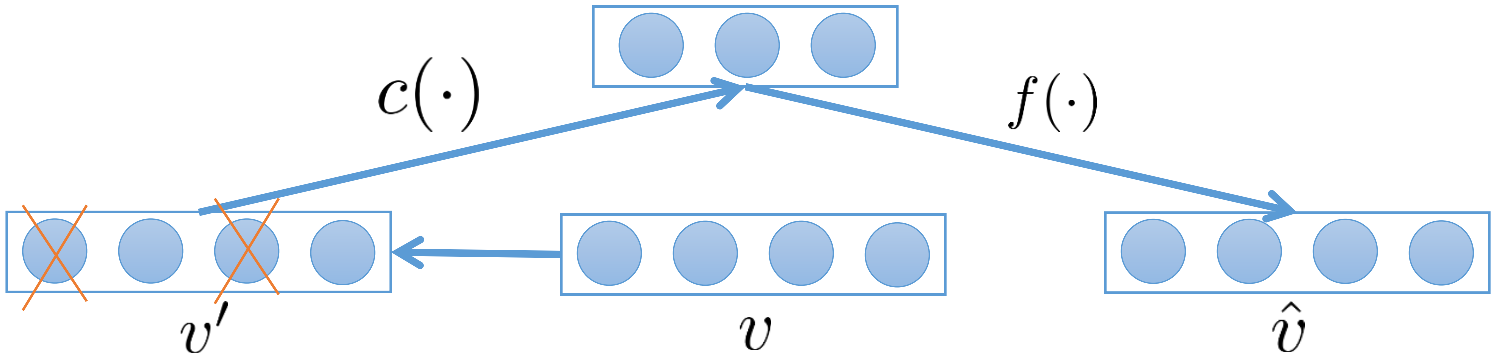
- In the following case
  - Binary units
  - The number of hidden units is equal to the number of visible units
- There is a trivial solution $W_1 = W_2 = I, \eta = \theta = -0.5$





0	-0.5	0	-0.5	0
1	0.5	1	0.5	1

Denoising auto-encoder

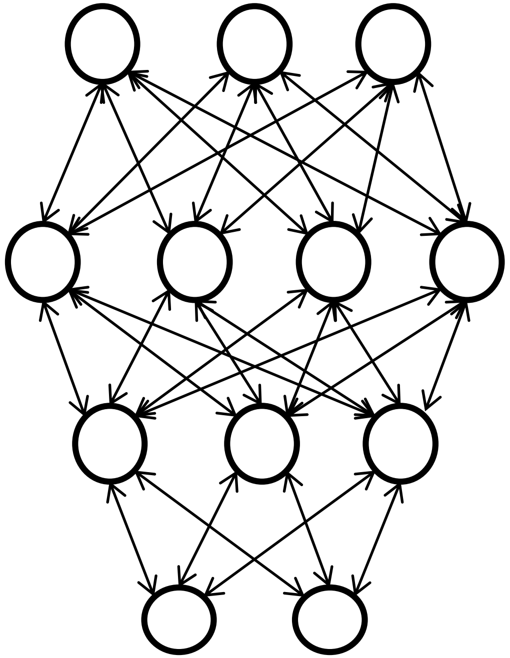


- Corrupt v to v' by randomly setting some elements of v to zero.
- Use v' as input and try to reconstruct v.
  - Ideally, $\hat{v}$ is the clean version of v

Vincent, et al., ICML, 2008

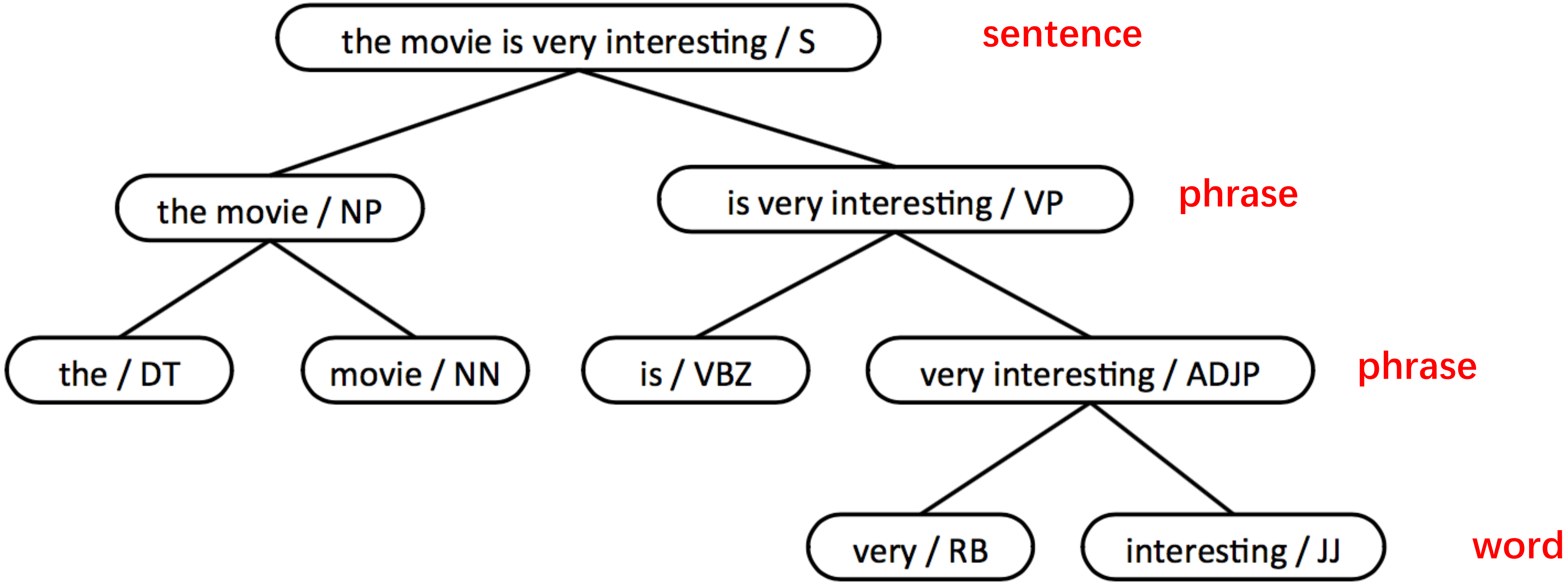
Deep Auto-encoder

- Stack auto-encoders on top of each other
- Train layers one by one
- Fine tune with BP
- Sparsity or other regularizations can be used



Recursive Autoencoders

Sentences/phrases have composition structures!



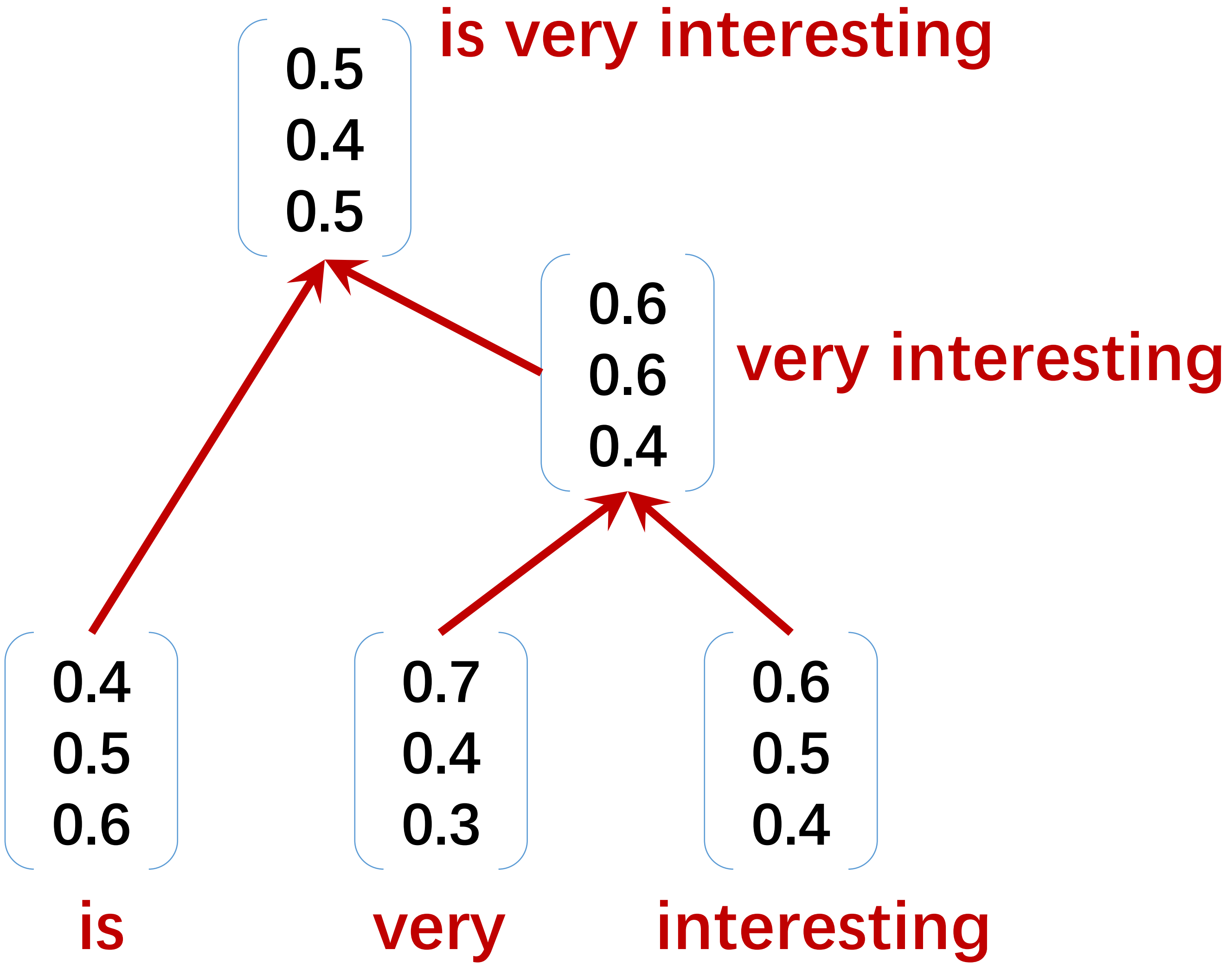
Recursive Autoencoders

Rules of Compositionality

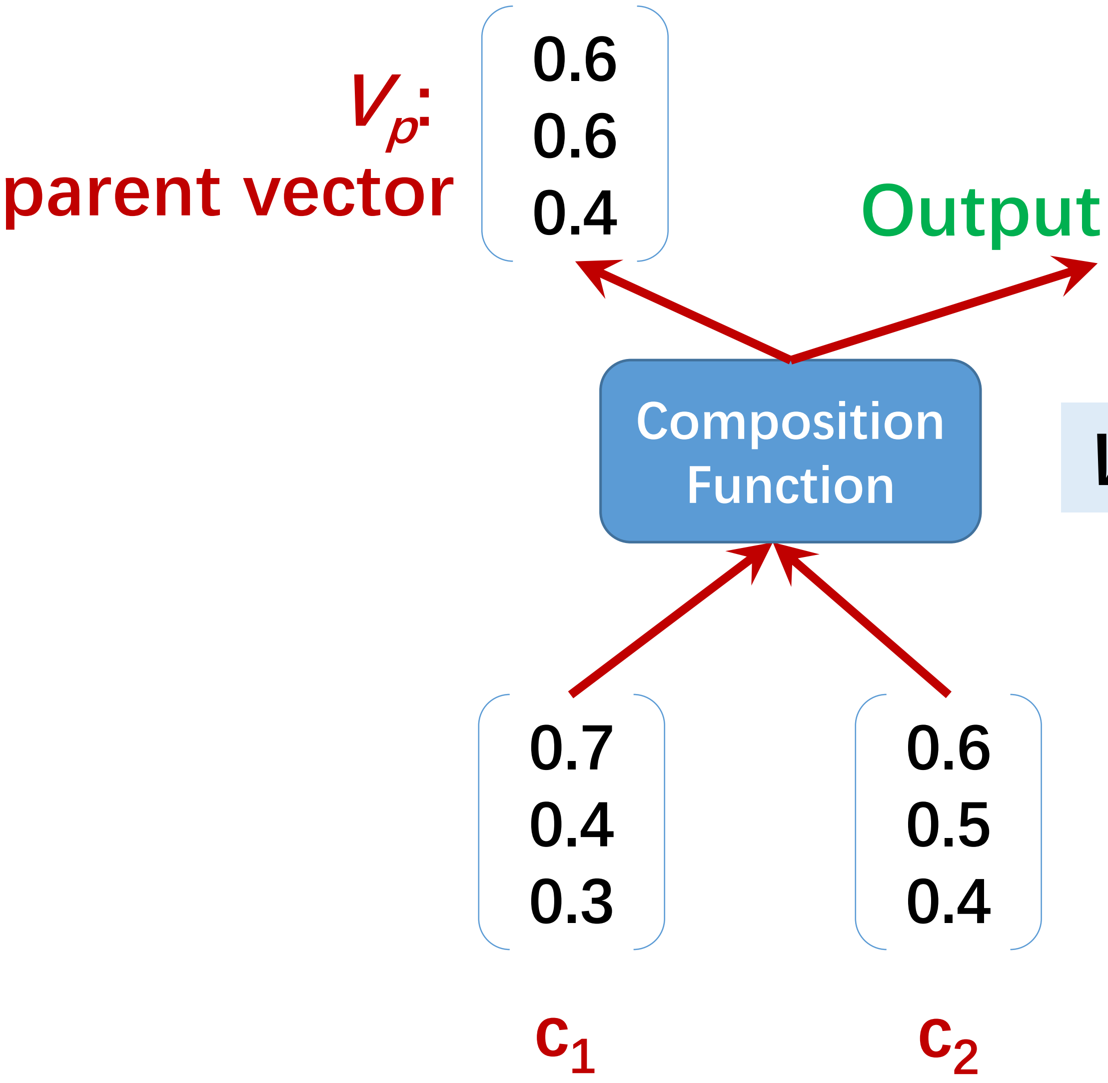
The meaning (vector) of a sentence is determined by

(1) **The meanings of its words**

(2) **The rules that combine them**



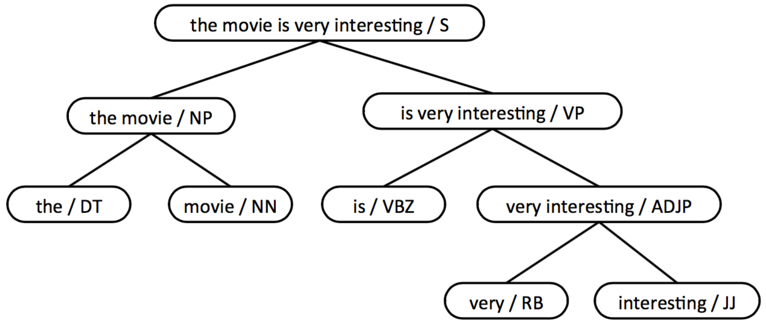
Recursive Autoencoders



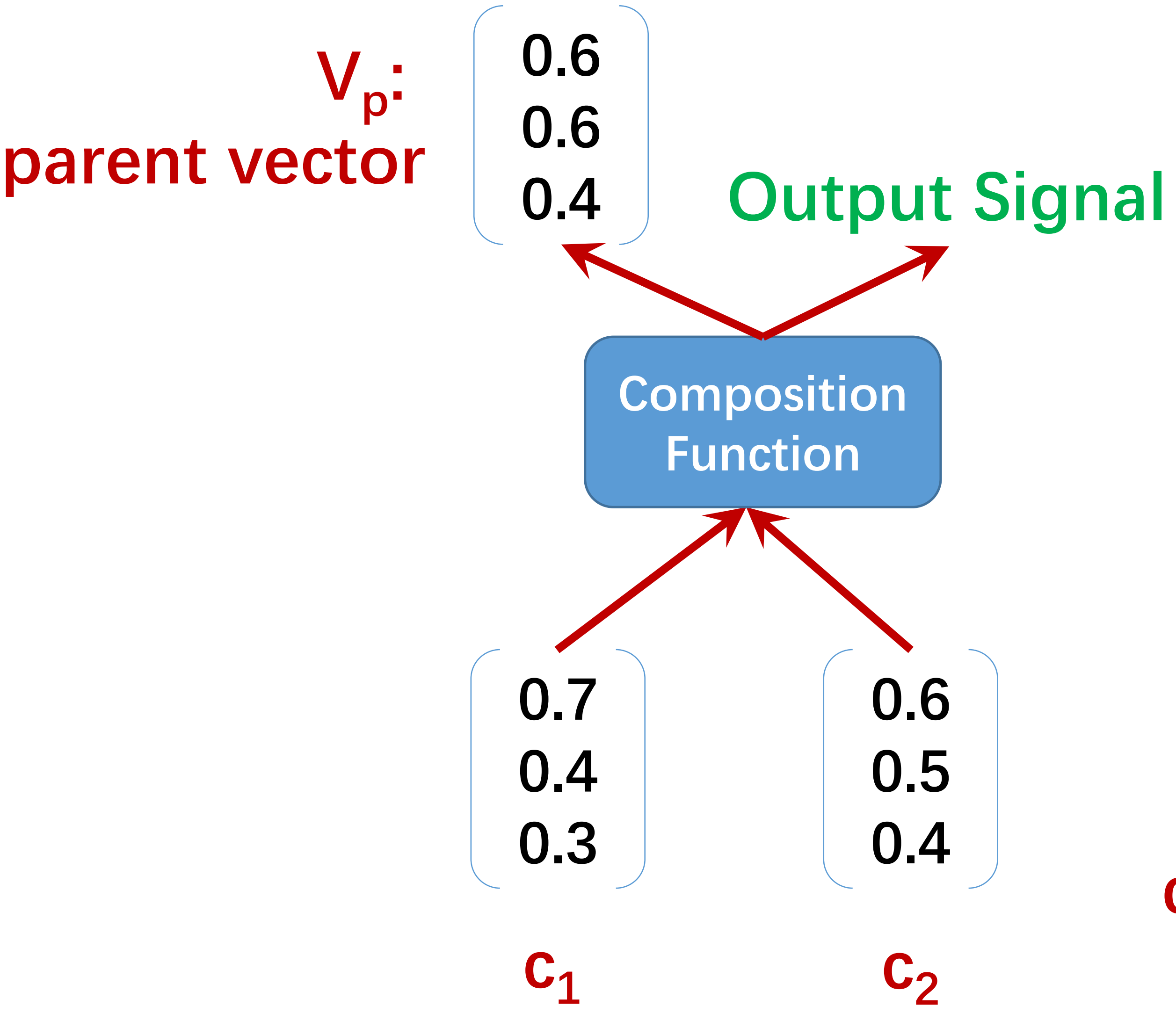
$\hat{p}$=softmax($U^T * V_p$)

$$Vp = \tanh\left(W\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + b\right)$$

Same *W* parameters at all nodes of the tree



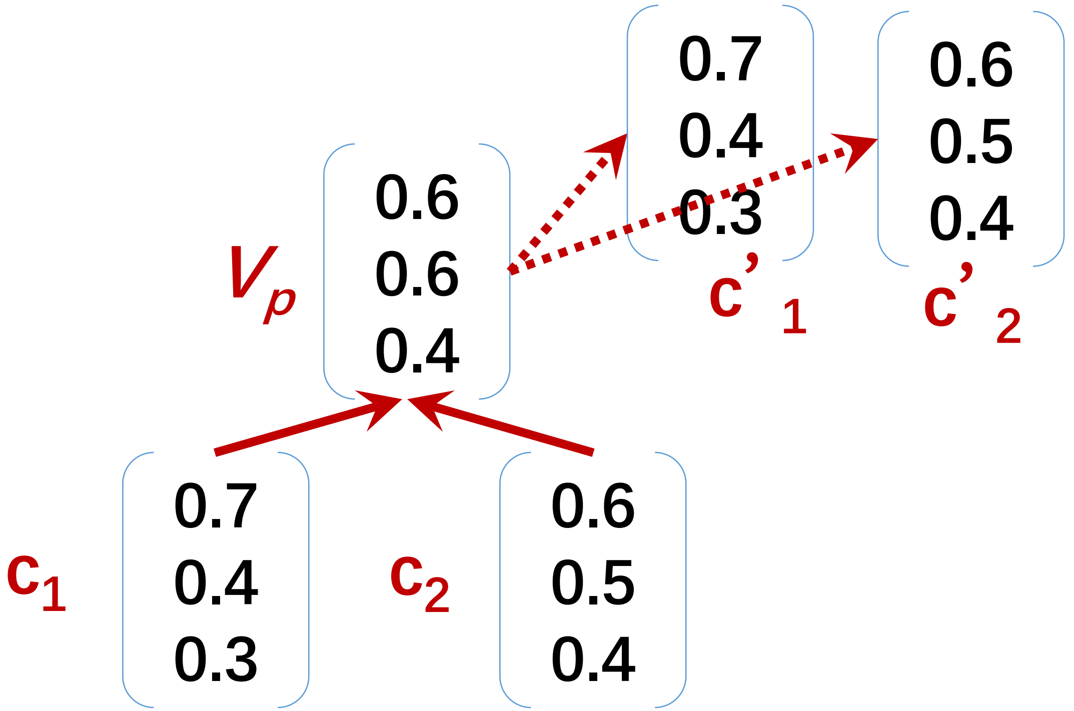
Recursive Autoencoders



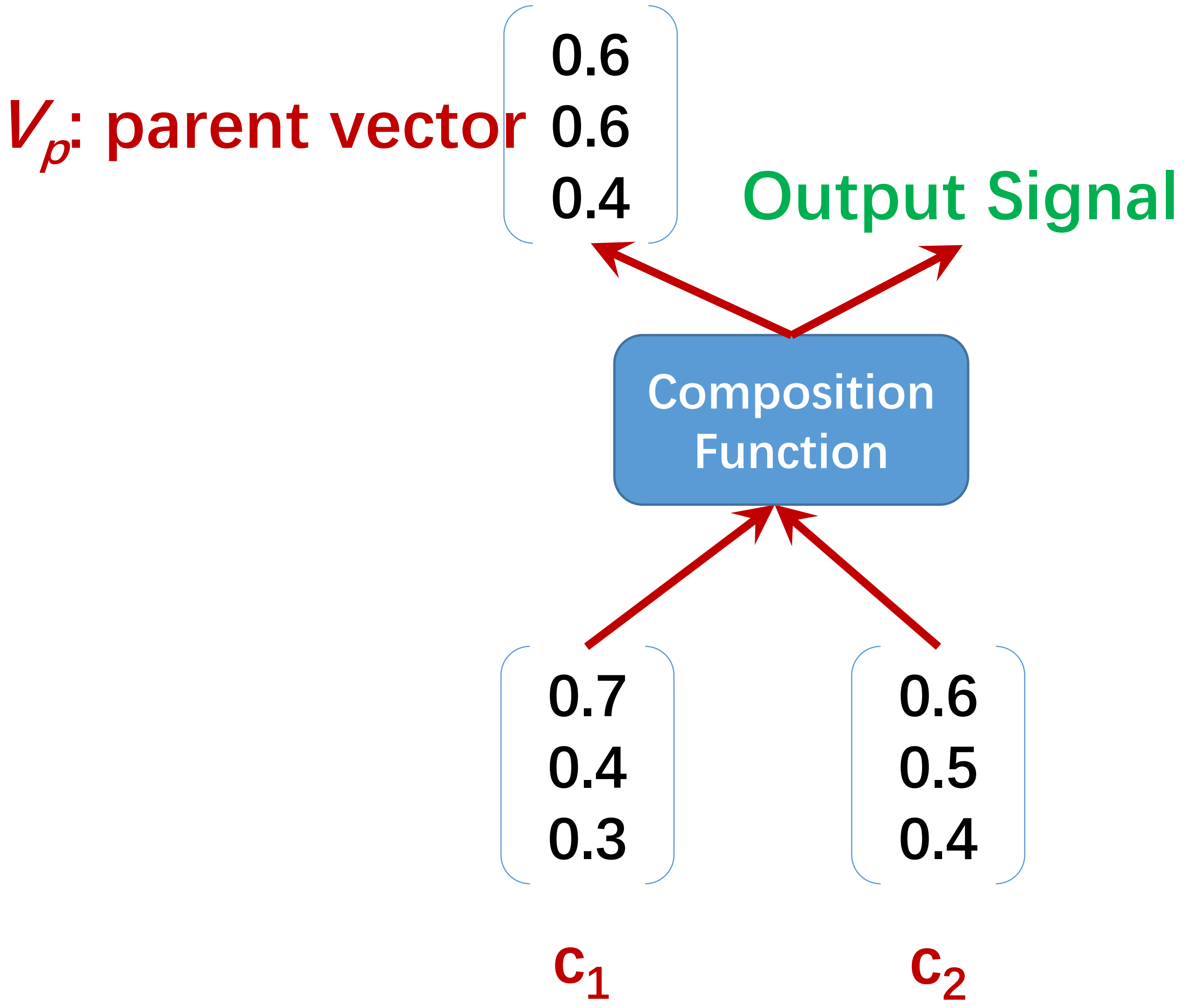
Train the model

✓ No supervision: minimizing reconstruction error

$$E_{rec}([c_1; c_2]) = \frac{1}{2}\left|\left|[c_1; c_2] - [c'_1; c'_2]\right|\right|^2$$



Recursive Autoencoders



Train the model

✓ With supervision:
minimizing cross entropy error

$$E = \sum_{k=1}^{K} -p(k) \log \hat{p}(k)$$

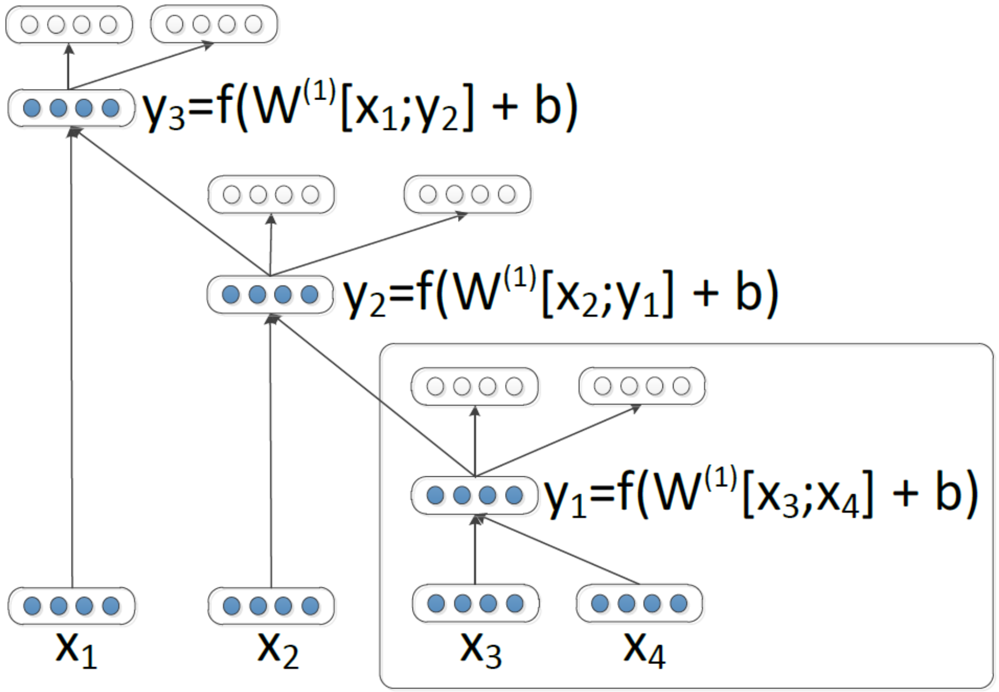
The gold distribution over class labels k

The predicted distribution based on the parent vector V_p

$$\hat{p} = \text{softmax}(U^T * V_p)$$

Recursive Autoencoder

- Given the tree structure, we can compute all the node vectors from bottom to top.
- Train by minimizing reconstruction error

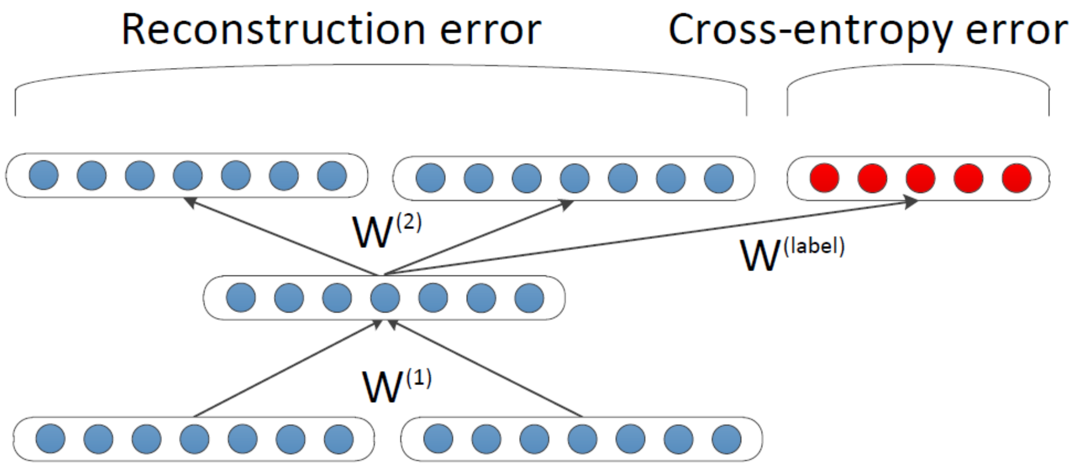


$$E_{rec}([c_1; c_2]) = \frac{1}{2} \left|\left|[c_1; c_2] - [c'_1; c'_2]\right|\right|^2$$

Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Semi-Supervised Recursive Autoencoders

- The task label can be introduced in all nodes of the tree.



$$d(p;\theta) = \text{softmax}(W^{label}p)$$

$$E_{cE}(p,t;\theta) = -\sum_{k=1}^{K} t_k \log d_k(p;\theta)$$

$$E([c_1;c_2]_s, p_s, t, \theta) =$$

$$\alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}(p_s,t;\theta)$$

Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Comments on Recursive Autoencoders

- Dependent on a tree structure
  - Parser is required
- Deep structures ($h = \log N$)
  - More supervision is required (at the internal nodes)