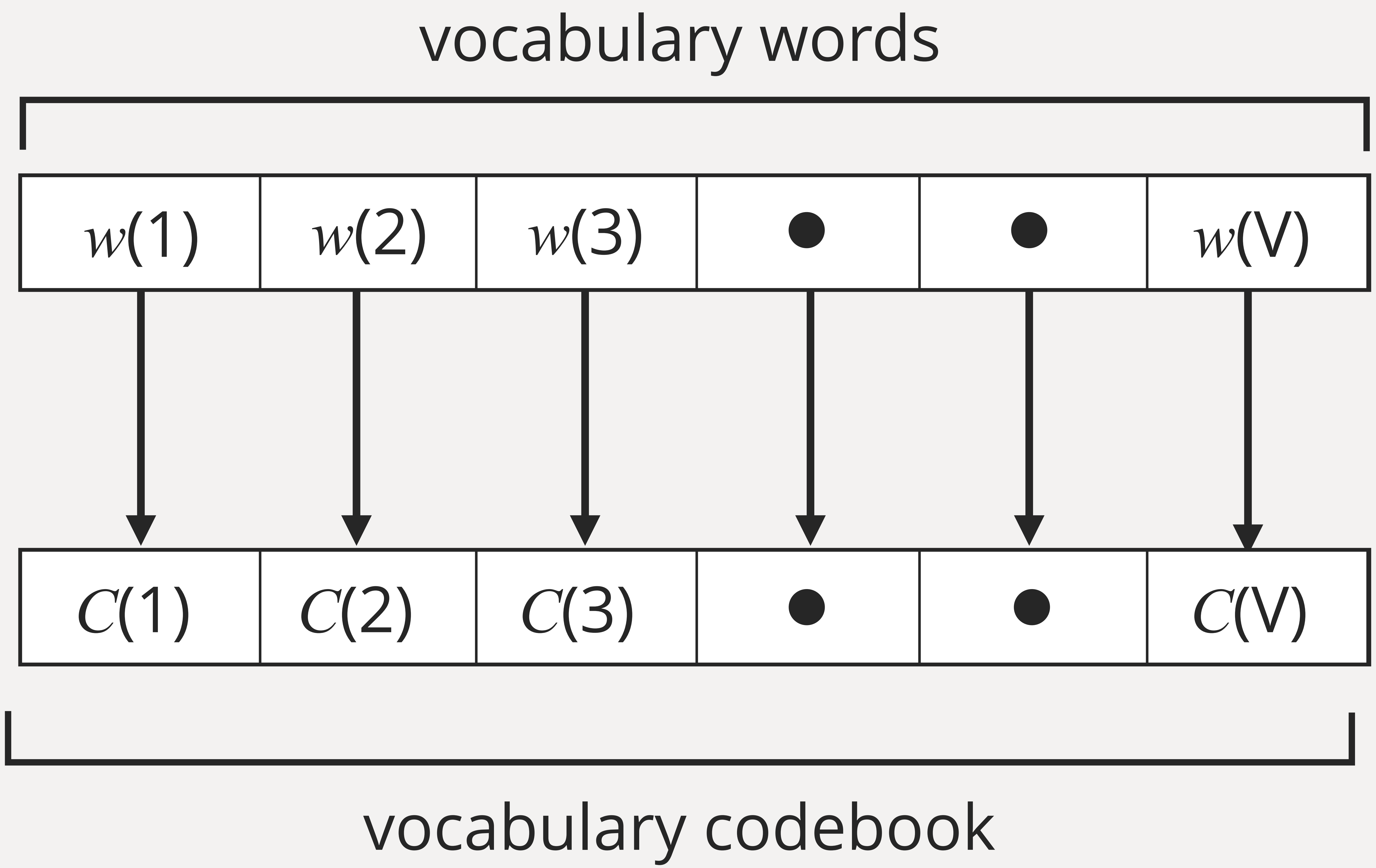
Neural Model of Text

Duke DataScience

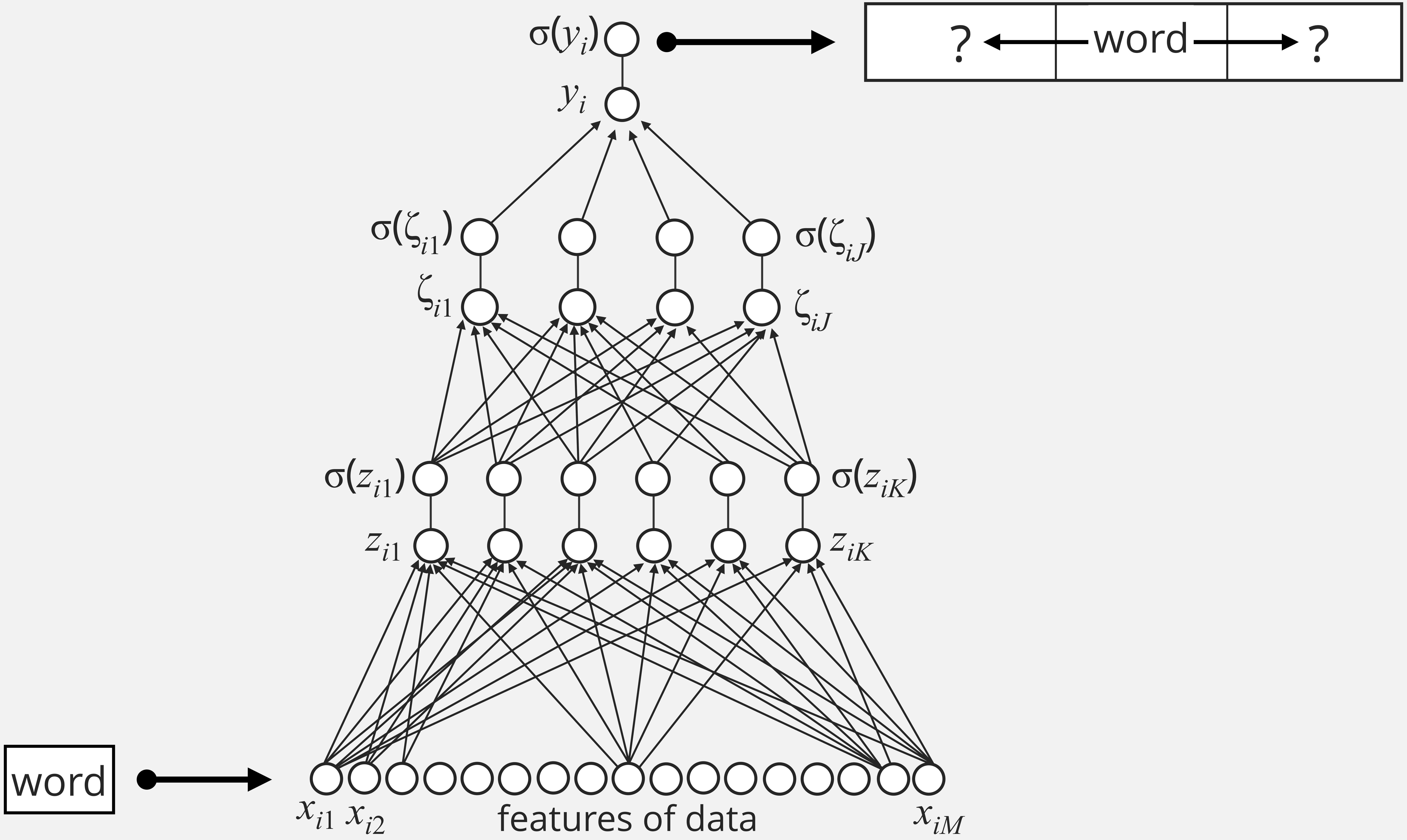
A codebook is a set of vectors where every word in the vocabulary is mapped to a vector

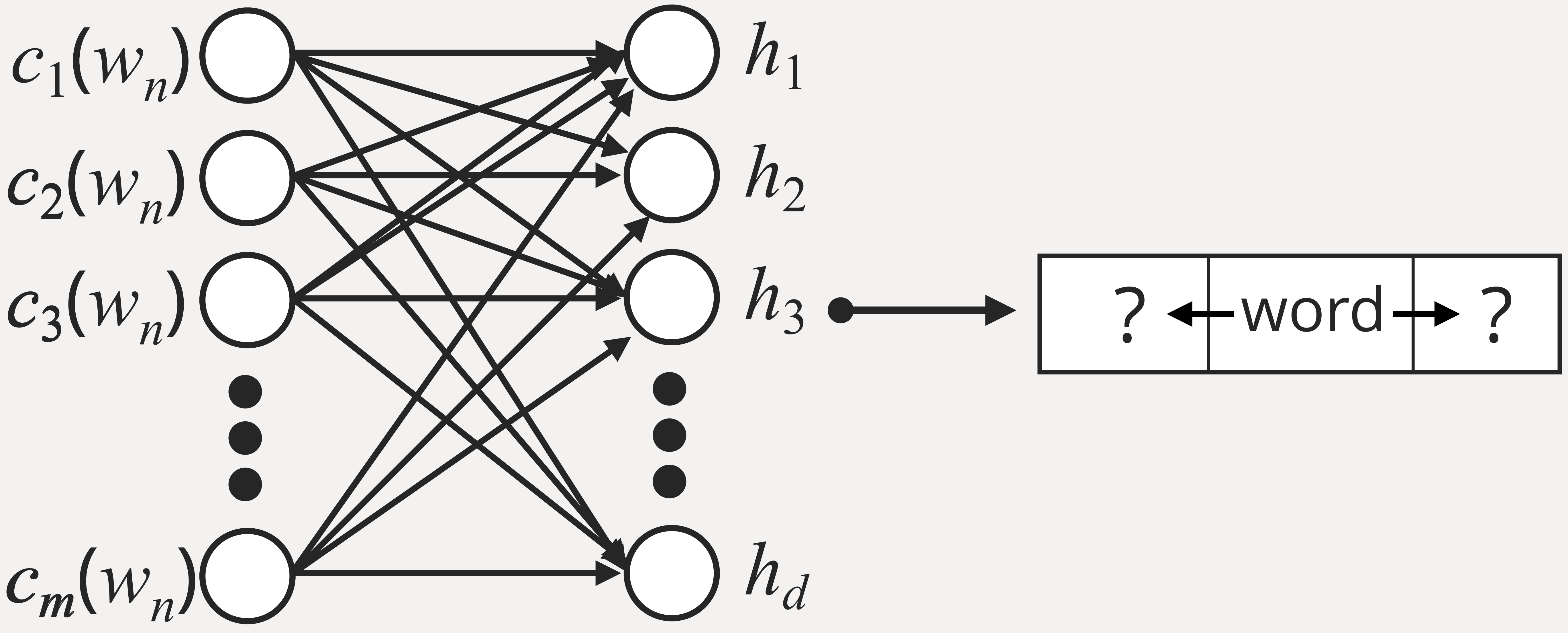


Learning the Codebook

- Would like to have unlabeled data
- Assume we have access to a large corpus of unlabeled documents

Model Prediction

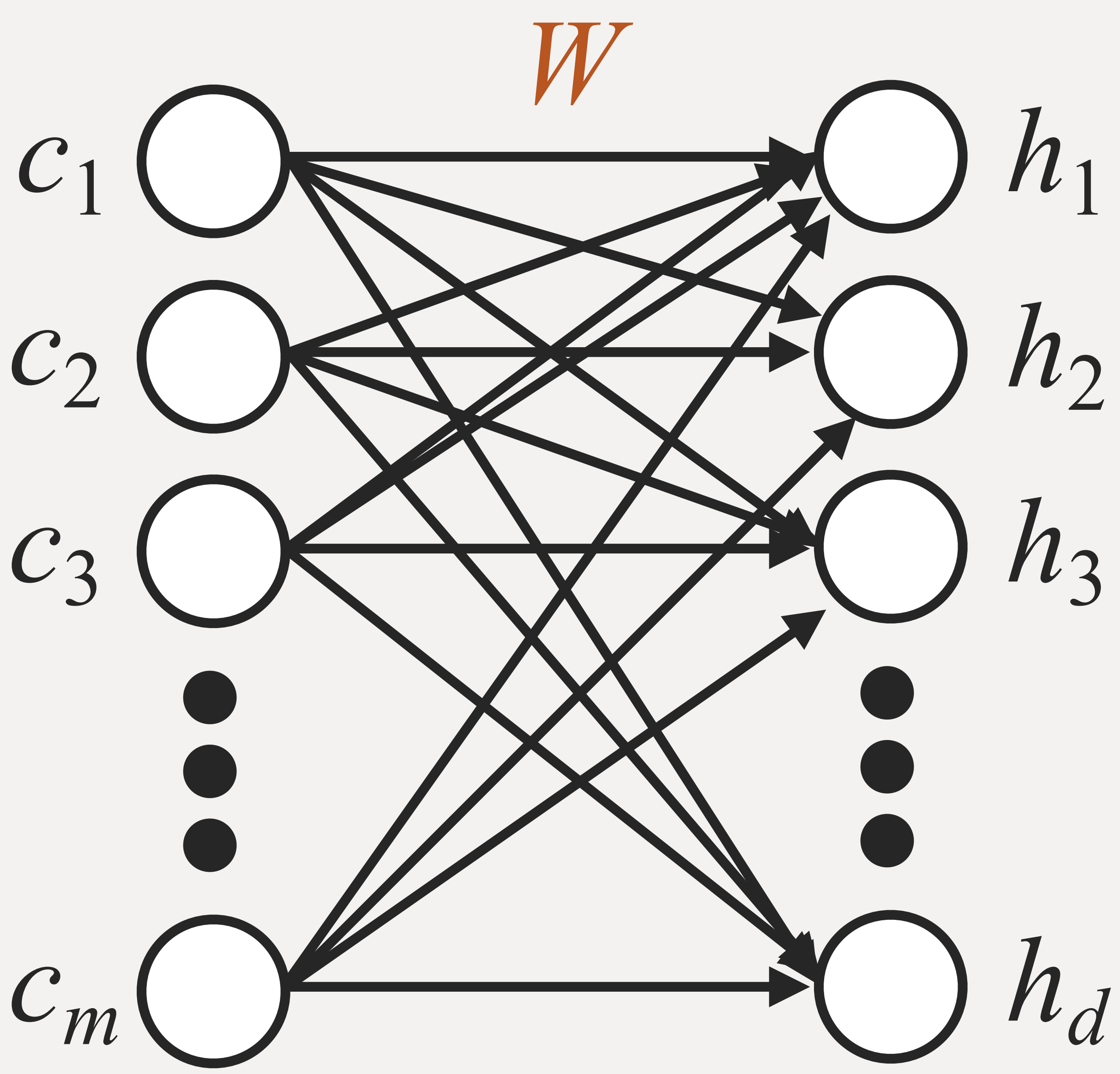




Learning Word Vectors

- $c(w_n)$ = m-dimensional code for w_n
- m-dimensional $c(\mathrm{w_n})$ = input to a multilayer perceptron
- Map to latent vector h that is d-dimensional
- Take d-dimensional vector to make prediction of surrounding words

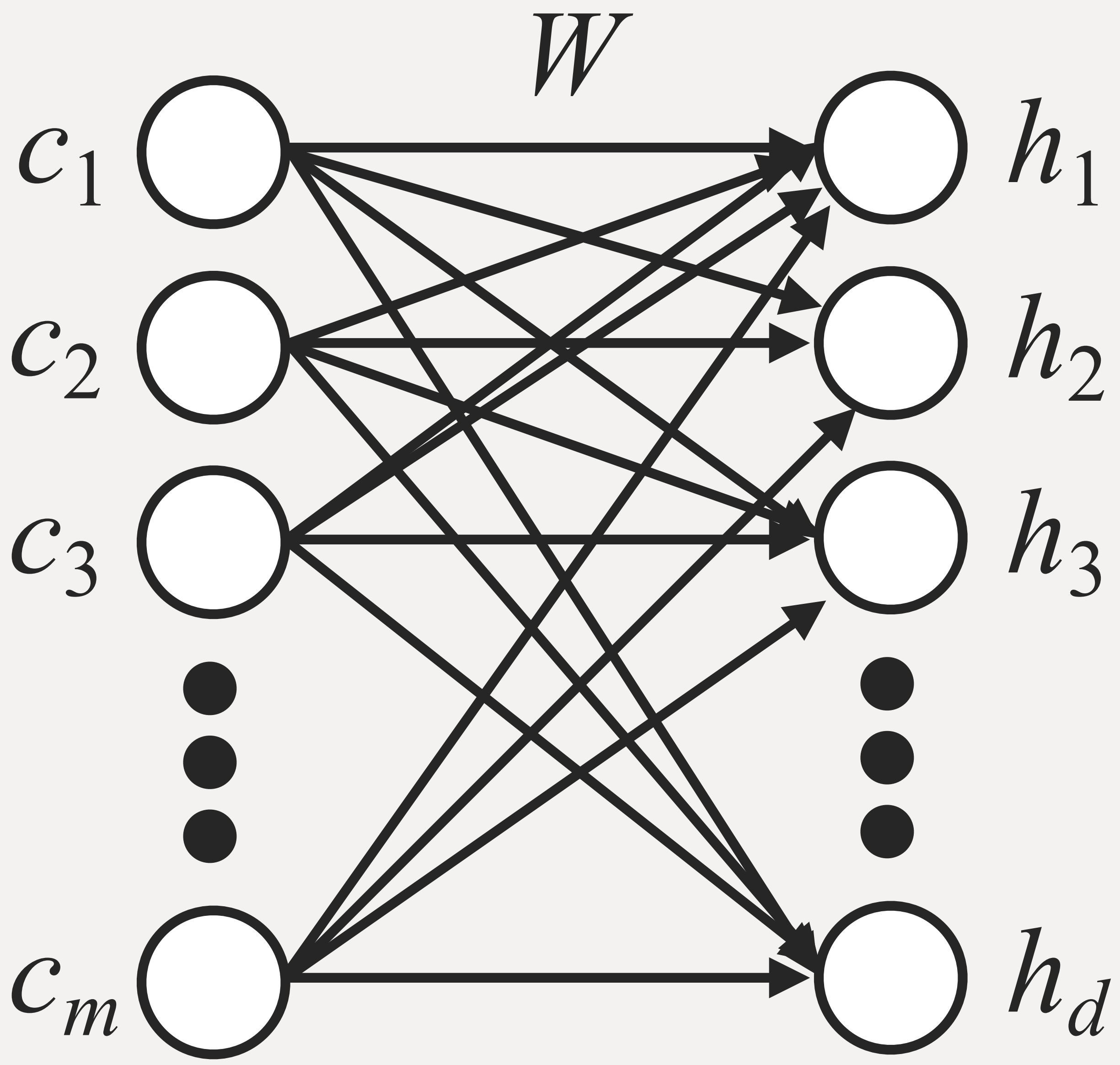
Mathematical Notation



$$h_j = g[W_{1j} \times c_1 + W_{2j} \times c_2 + \cdots + W_{mj} \times c_m + b_j]$$

weights

Mathematical Notation



$$h_j = g[W_{1j} \times c_1 + W_{2j} \times c_2 + \cdots + W_{mj} \times c_{\mathrm{m}} + b_j]$$

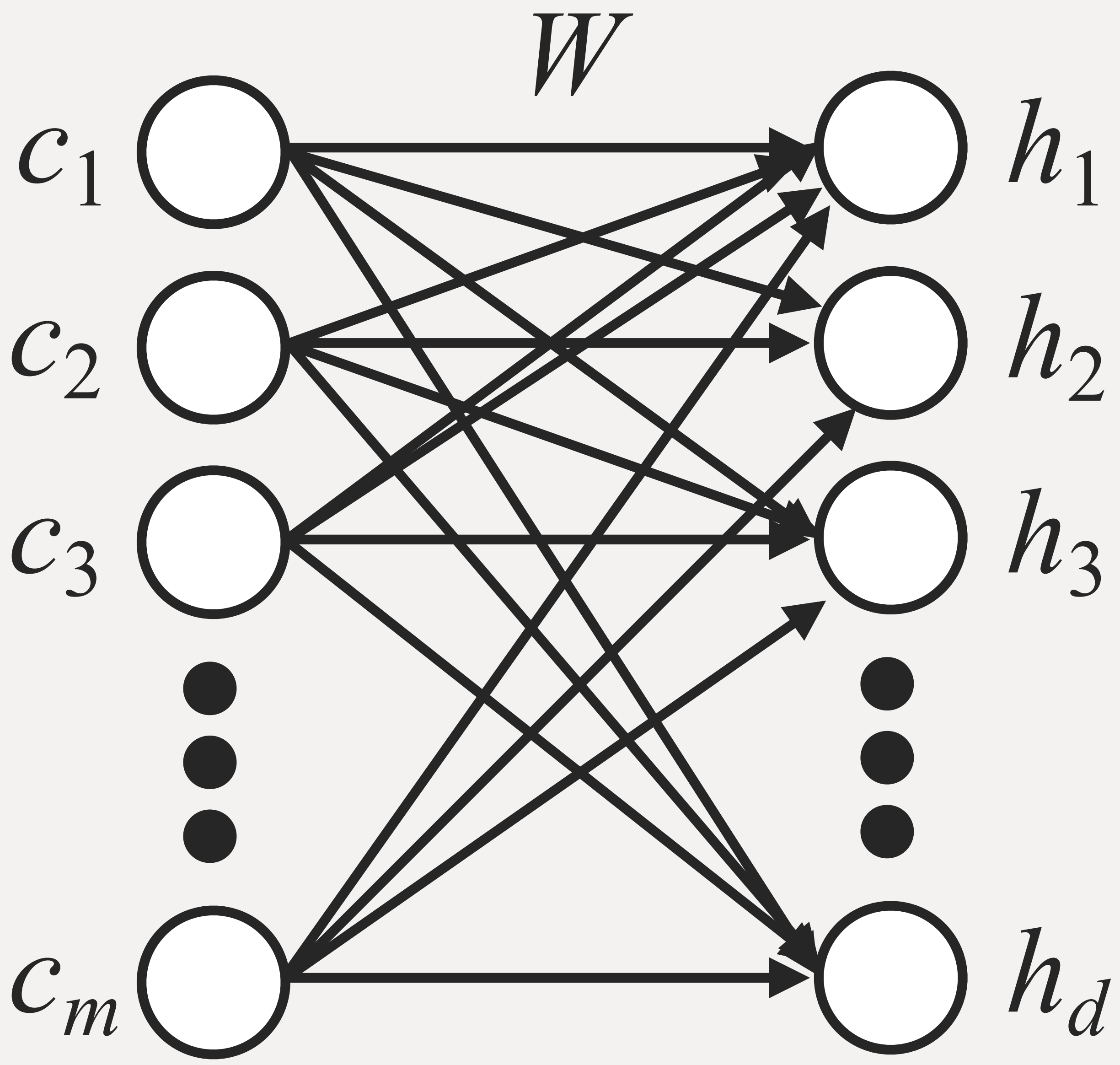
nonlinear function

concise notation

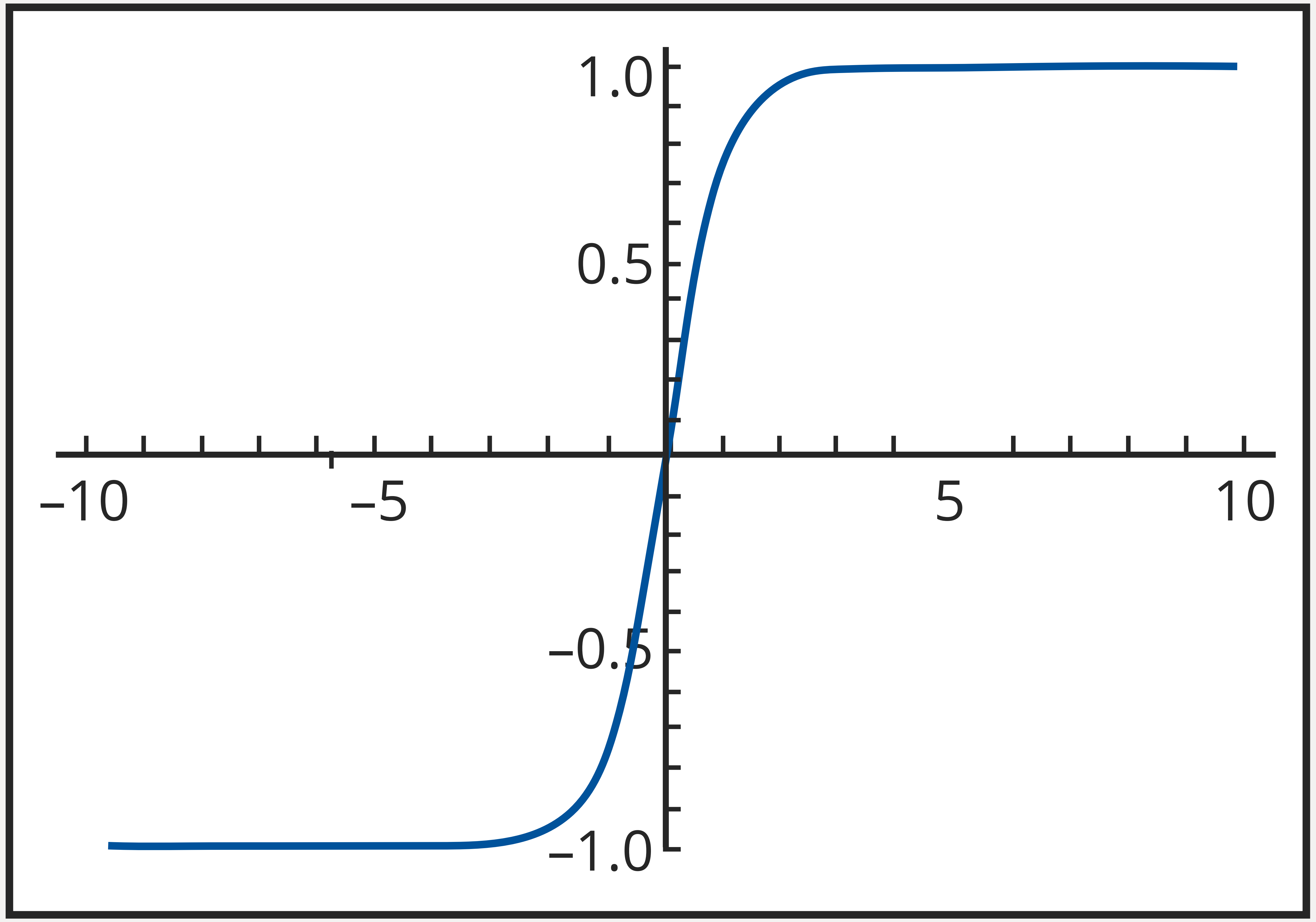
bias

$$h = g[W \times c + b]$$

$$h = \tanh[\overbrace{W \times c + b}^{\text{vector}}]$$

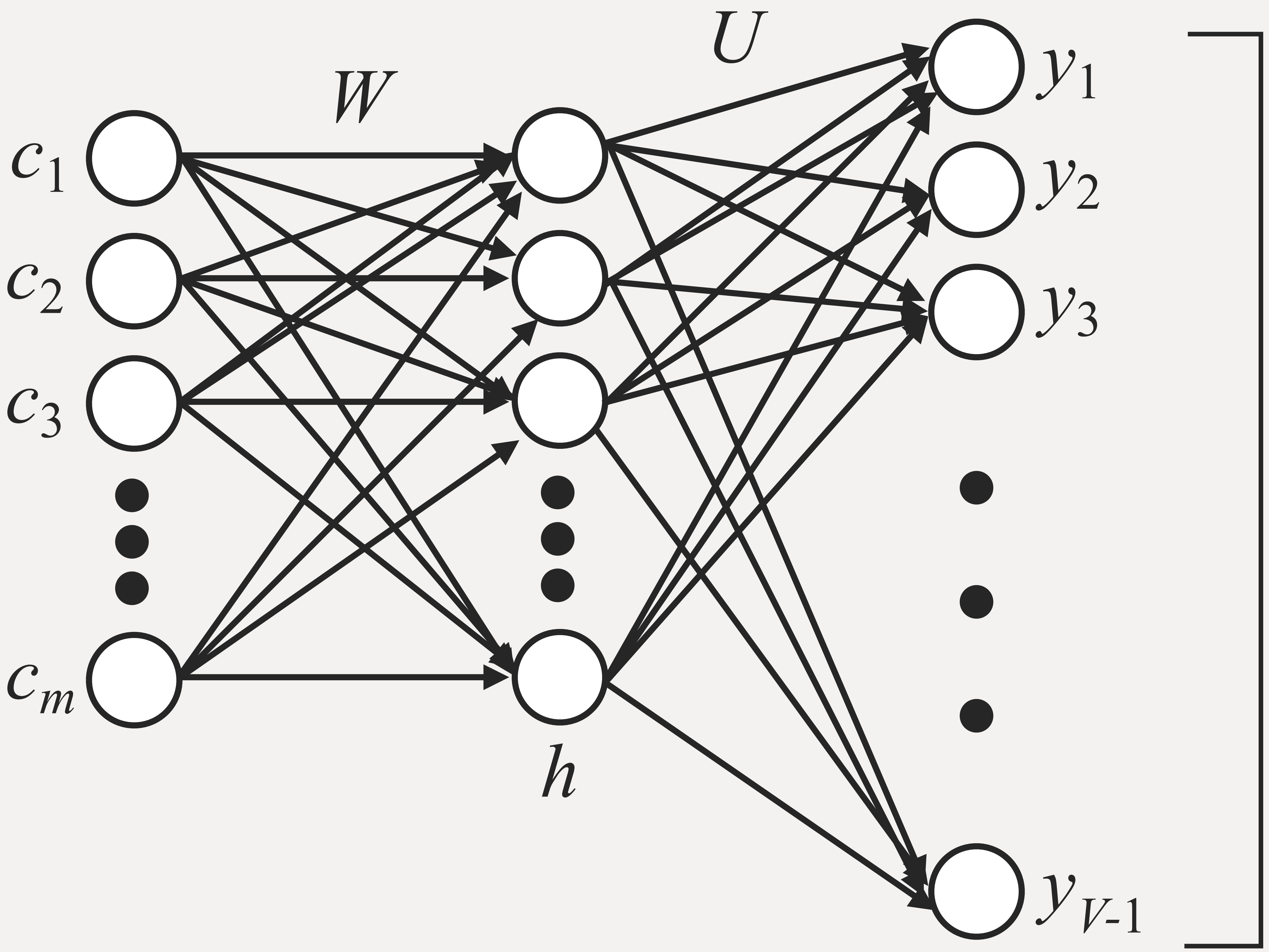


$g(x) = \tanh(x)$



tanh[W x c + b] operates pointwise or separately on components of the vector

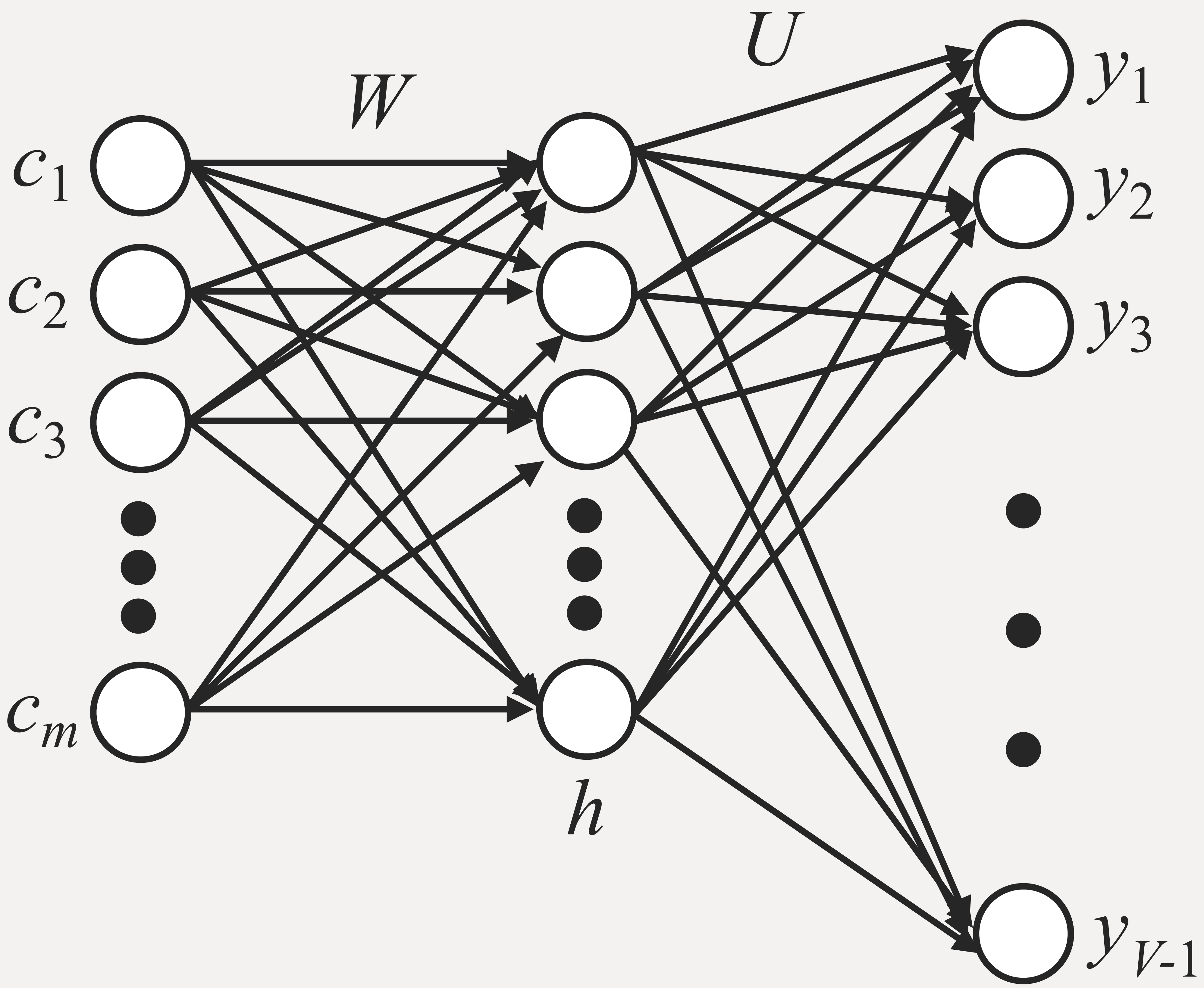
Hidden Units to Words



output dimension = V–1

V = size of vocabulary

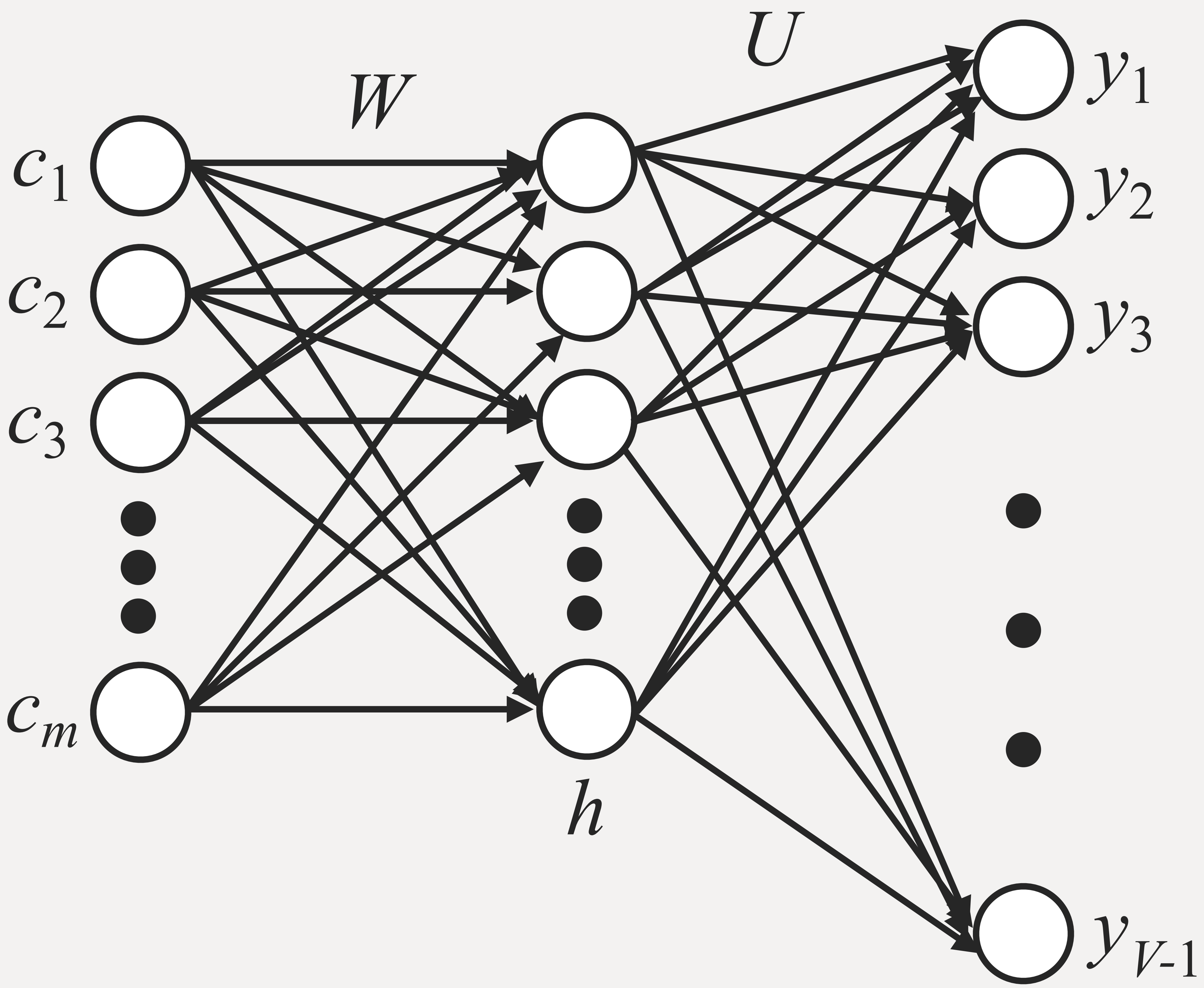
Hidden Units to Words



$$y_v = U_{1v} \times h_1 + U_{2v} \times h_2 + \cdots + U_{dv} \times h_d + \beta_v$$

weights

Hidden Units to Words



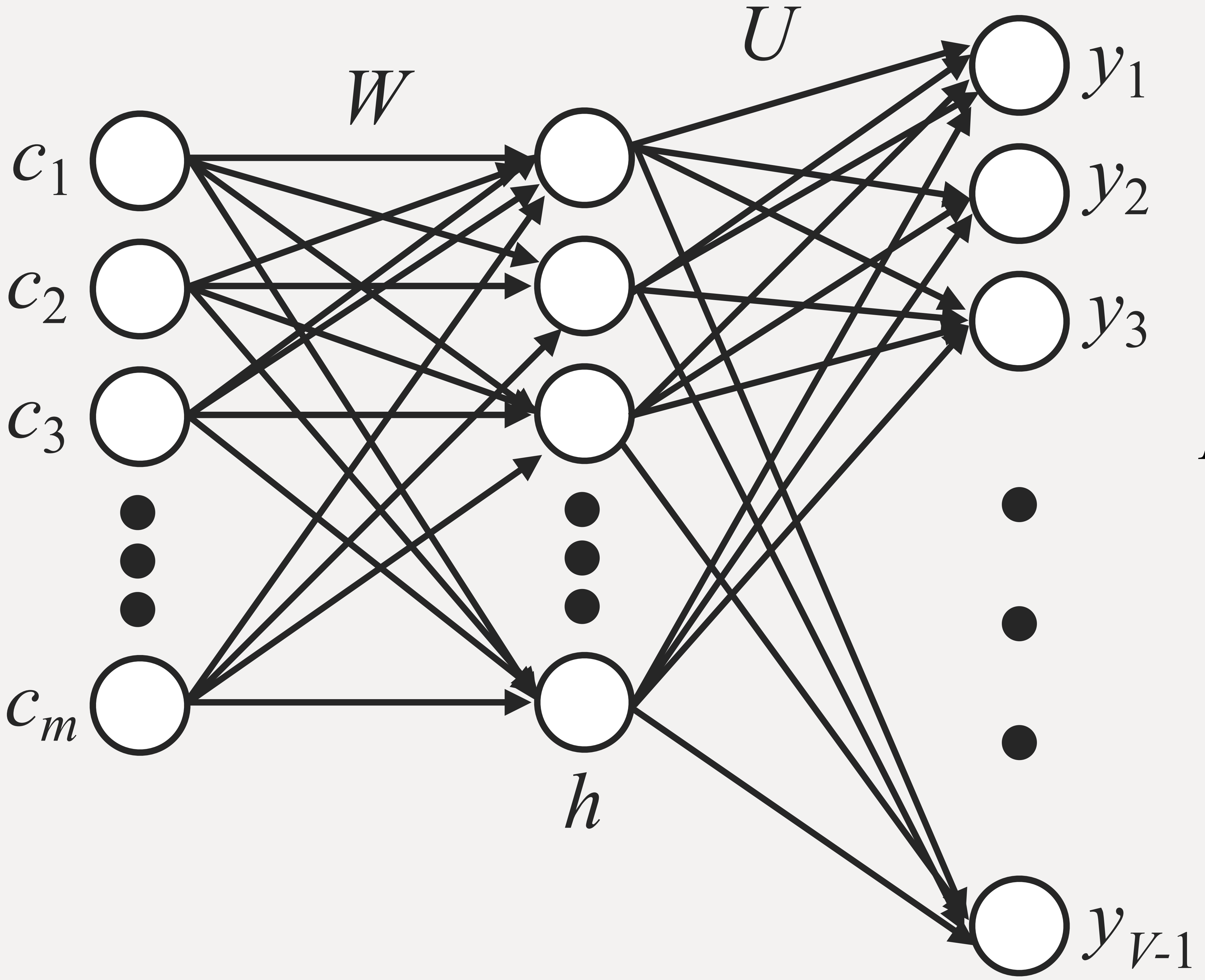
$$y_v = U_{1v} \times h_1 + U_{2v} \times h_2 + \cdots + U_{dv} \times h_d + \beta_v$$

bias

concise notation

$$y = U \times h + \beta$$

Generalize Logistic Regression



Softmax

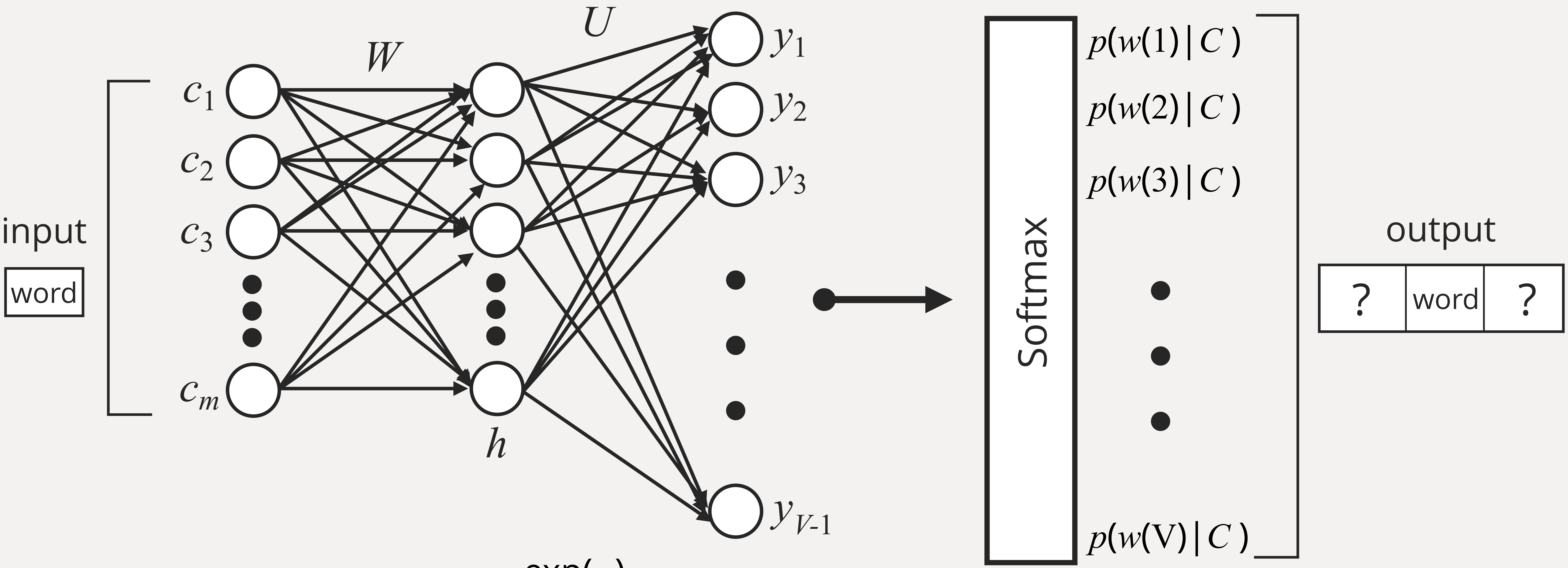
n^{th} word

$$p(w(i) \mid C(w_n)) = \frac{\exp(y_i)}{\exp(y_1) + \exp(y_2) + \cdots + \exp(y_V)}$$

i^{th} word

y_V set to 0

Neural Model for Text



$$p(w(i)\,|\,C\,) = \frac{\exp(y_i)}{\exp(y_1) + \exp(y_2) + \cdots + \exp(y_{V\text{-}1}) + 1}$$

Word2Vec