

VL Deep Learning for Natural Language Processing

10. Named Entity Recognition

Prof. Dr. Ralf Krestel
AG Information Profiling and Retrieval

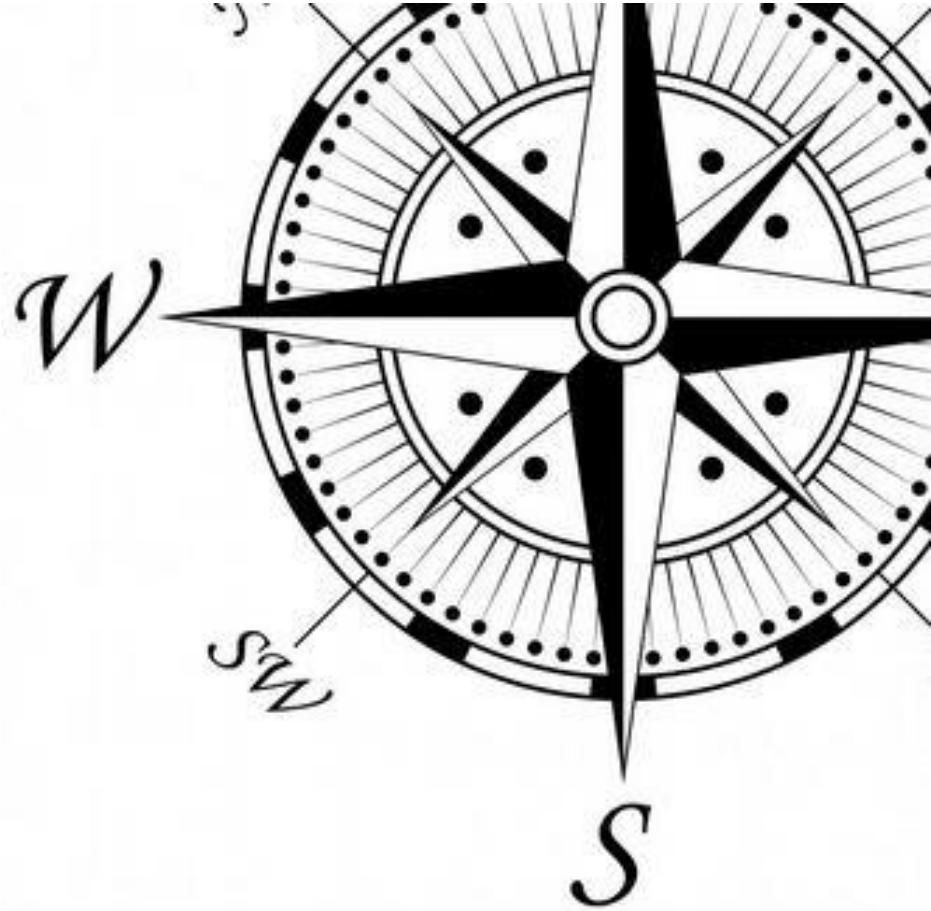
Learning Goals for this Chapter



- Know different input representations for text
 - and how/when to use them
 - Develop a simple window classifier
 - Understand the max-margin loss function
 - and know how to employ it
-
- Relevant chapters
 - P6.1
 - S3 (2019) <https://www.youtube.com/watch?v=PLryWeHPcBs>

Topics Today

1. **Text as Input to Neural Networks**
2. DNN to Classify Words
3. The Max-Margin Loss Function



What Makes Text Representation Hard?



- Noise
 - Spelling mistakes, colloquialisms, slang words, emoticons, ...
- Ambiguity
 - multiple meanings, sarcasm, ...
- Semantic structure
 - order and placement of words can drastically change the meaning, e.g., “not”
- Domain knowledge
 - understanding of the domain and domain jargon necessary
- Multilingualism
 - Mixed-code text, low resource languages

How to Process Text?

- Token-level
 - Character by character
- Word-level
 - One word at a time
- Sentence-level
 - Or fixed-size context window
- Document-level
 - Fixed-length chunk
 - E.g., first 100 words

Typically: sequential processing



- Recurrent neural networks (LSTM, GRU)
- Seq2seq models
- Transformers

Typically fixed-length, non-sequential



- Feed forward neural networks
- Convolutional neural networks (CNN)
- Generative adversarial networks (GAN)

How to Represent Text?

Alternatively: multiset of
1-hot-encoded word vectors



- **Bag-of-words (BOW) model**

- Keep the frequency: word count
- Other weights, e.g., TF-IDF
- Binary: set of words, incident vectors
(0/1 whether a word occurs at least once)
- Optional: smoothing
- Optional: normalization, stemming, etc.
- Optional: n-grams

- Example

DOC1: *John likes dogs. Mary likes cats. Both play soccer.*

dogs		DOC1
0	John	1
0	likes	2
1	dogs	1
0	Mary	1
0	cats	1
0	Both	1
0	play	1
0	soccer	1
0	...	0
⋮	⋮	⋮
0	Zyzzzyva	0

How to Represent Text?



- **Topic model**

- Learn a latent (hidden) space
- Based on singular value decomposition (SVD)
 - Latent semantic analysis (LSA) aka latent semantic indexing (LSI)
- Based on probabilistic models
 - Probabilistic LSA (PLSA)
 - Latent Dirichlet allocation (LDA)

		Manual topic titles	
dogs			DOC1
$\begin{bmatrix} 0.8 \\ 0 \\ 0.1 \\ 0 \\ \vdots \\ 0.1 \end{bmatrix}$	<i>nature</i>	$\begin{bmatrix} 0.4 \\ 0.1 \\ 0.4 \\ 0 \\ \vdots \\ 0.1 \end{bmatrix}$	
	<i>politics</i>		
	<i>sport</i>		
	...		
	\vdots		
	<i>entertain</i>		

- Example

DOC1: *John likes dogs. Mary likes cats. Both play soccer.*

Dimensions in
latent space

How to Represent Text?



- **Embedding model**

- Character-level
- Subword-level
- **Word-level**
- Phrase- and sentence-level
- Paragraph-level
- Document-level
- Combinations of levels also possible
 - e.g. character + word for NMT
 - or using lower-levels as additional input

} Most meaningful

Can also be context-dependent

Dimensions in latent space

dogs
 $\begin{bmatrix} 0.16 \\ 0.21 \\ 0.18 \\ 0.04 \\ \vdots \\ 0.10 \end{bmatrix}$

Meaningful structure along arbitrary directions

DOC1
 $\begin{bmatrix} 0.05 \\ 0.11 \\ 0.09 \\ 0.02 \\ \vdots \\ 0.04 \end{bmatrix}$

Average or sum of word vectors

Alternatively: Multiset of word vectors

DOC1: *John likes dogs. Mary likes cats. Both play soccer.*

Most Tasks and Applications



- Ordered multiset of words
 - 1-hot-encoded



- Ordered multiset of embeddings
 - Dense word vectors for each word
 - Can be pretrained/fine-tuned

- Sequence of words
 - 1-hot-encoded

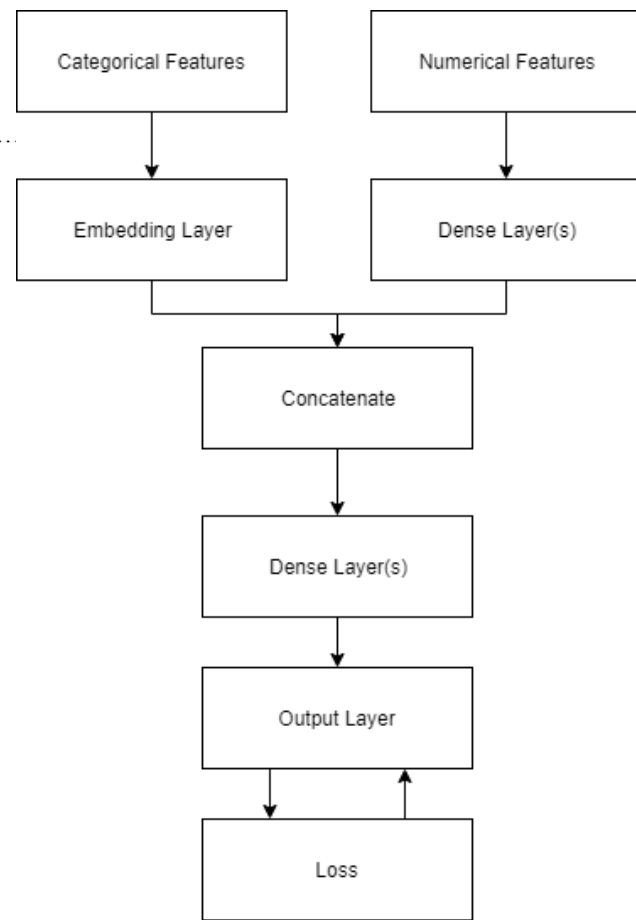


- Sequence of embeddings
 - Dense word vectors for each word
 - Can be pretrained/fine-tuned

Different Input Data Types

- Categorical
 - Character
 - String (=array of char)
 - Text (indexed strings)
- Numerical
 - Real
 - Integer
- Combining different feature types
 - Concatenating their representations

Objects
with IDs



The Embedding Layer

- Example: DOC2: *John likes dogs. Mary likes cats.*
- Index of words: $[0,1,2,3,1,4]^T$
- Input matrix I:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Why do we need a special embedding layer?
Why not use a normal fully connected, dense layer?

- Assume a 3-dimensional embedding layer with weight matrix W:

$$\begin{bmatrix} 0.1 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \\ 0.5 & 0.6 & 0.5 \\ 0.3 & 0.2 & 0.8 \\ 0.6 & 0.6 & 0.1 \end{bmatrix}$$

The Embedding Layer



- The output would be $I * W =$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.1 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \\ 0.5 & 0.6 & 0.5 \\ 0.3 & 0.2 & 0.8 \\ 0.6 & 0.6 & 0.1 \end{bmatrix}$$

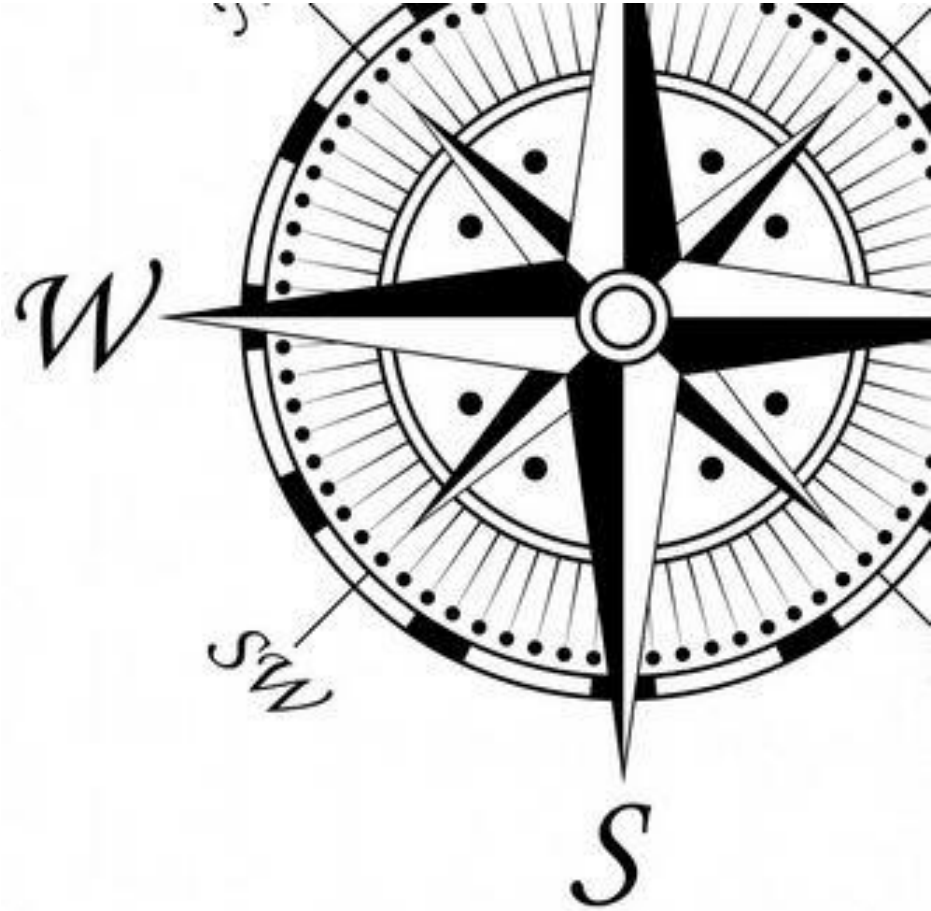
$$= \begin{bmatrix} 1 * 0.1 + 0 * 0.4 + 0 * 0.5 + 0 * 0.3 + 0 * 0.6 & 1 * 0.4 + 0 * 0.3 + 0 * 0.6 + 0 * 0.2 + 0 * 0.6 & 1 * 0.2 + 0 * 0.2 + 0 * 0.5 + 0 * 0.8 + 0 * 0.1 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.4 & 0.2 \\ 0.4 & 0.3 & 0.2 \\ 0.5 & 0.6 & 0.5 \\ 0.3 & 0.2 & 0.8 \\ 0.4 & 0.3 & 0.2 \\ 0.6 & 0.6 & 0.1 \end{bmatrix}$$

The embedding layer selects directly the right vectors: $[0,1,2,3,1,4]^T$

Topics Today

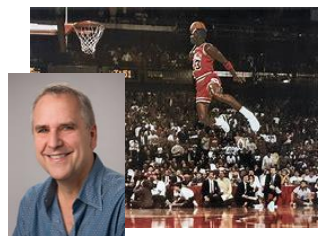
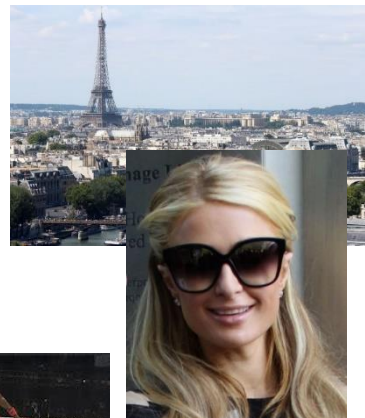
1. Text as Input to Neural Networks
2. **DNN to Classify Words**
3. The Max-Margin Loss Function



Word Classification



- Isolated, individual words are rarely classified
 - Meaning of words only clear in context
- Ambiguity can only be resolved in context
 - Autoantonyms: Same words, opposite semantics
 - Overlook, comprise, dust, left, ...
 - Ambiguous named entities
 - Paris
 - Michael Jordan
 - Orange



Classification with Context



- Idea: Classification of a word in the context of its k neighboring words
- E.g. named entity recognition (NER) can be modeled as word classification with four classes:
 - person, organization, location, none
- Naive approach:
 - Average of all word vectors in the context window.
 - Problem: Position information is lost
- Better:
 - Concatenating all word vectors from the context window

- Softmax classifier
 - Input: the target word to be classified and neighboring words in a context window of size k , left and right of the target word
 - Output: probability distribution over four named entity classes
- Example: Classification of the word *Paris* as location with context of size two:

...	museums	in	Paris	are	amazing	...
	$\begin{bmatrix} \blacklozenge \\ \blacklozenge \\ \blacklozenge \end{bmatrix}^T$	$\begin{bmatrix} \blacklozenge \\ \blacklozenge \\ \blacklozenge \end{bmatrix}^T$	$\begin{bmatrix} \blacklozenge \\ \blacklozenge \\ \blacklozenge \end{bmatrix}^T$	$\begin{bmatrix} \blacklozenge \\ \blacklozenge \\ \blacklozenge \end{bmatrix}^T$	$\begin{bmatrix} \blacklozenge \\ \blacklozenge \\ \blacklozenge \end{bmatrix}^T$	
$x_{window} =$	$[x_{museums}$	x_{in}	x_{Paris}	x_{are}	$x_{amazing}]^T$	

- The input vector is a column vector in \mathbb{R}^{5d}
 - $x_{window} = [\blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge \blacklozenge]^T$

Simplest Window Classifier



- Use softmax function!
- With $x = x_{window}$ we can use softmax on the concatenated word vectors

$$\hat{y}_y = p(y|x) = \frac{\exp(W_y \cdot x)}{\sum_{c=1}^C \exp(W_c \cdot x)}$$

$$W \in \mathbb{R}^{C \times d}$$

- Loss function as usual: cross entropy

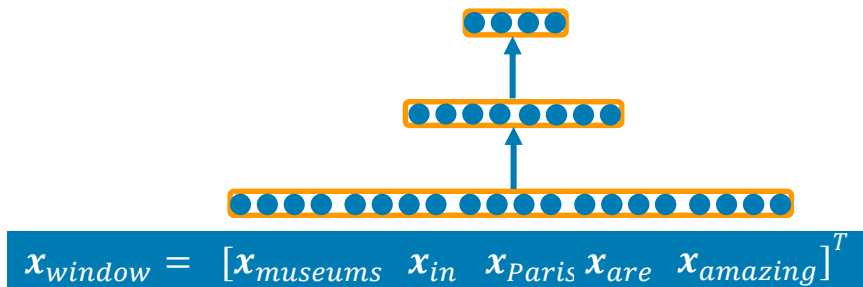
$$J(\theta) = \frac{1}{N} \sum_{i=1}^N -\log \left(\frac{e^{f_{y_i}}}{\sum_{c=1}^C e^{f_c}} \right)$$

$$f_y = W_y \cdot x$$

- How are the word vectors learned?
 - Gradient descend!
 - Analogous to word2vec

Additional Layer

- Up to now, we only have an input layer (word vectors) and a softmax output layer.
- Softmax can only discriminate linearly in the input space
 - No dependencies of the individual words in the window
- Thus, additional layer with non-linear activation function
 - Example: only if „*museums*“ is the first vector, then the „*in*“ at position two is important



Single Layer Neural Network

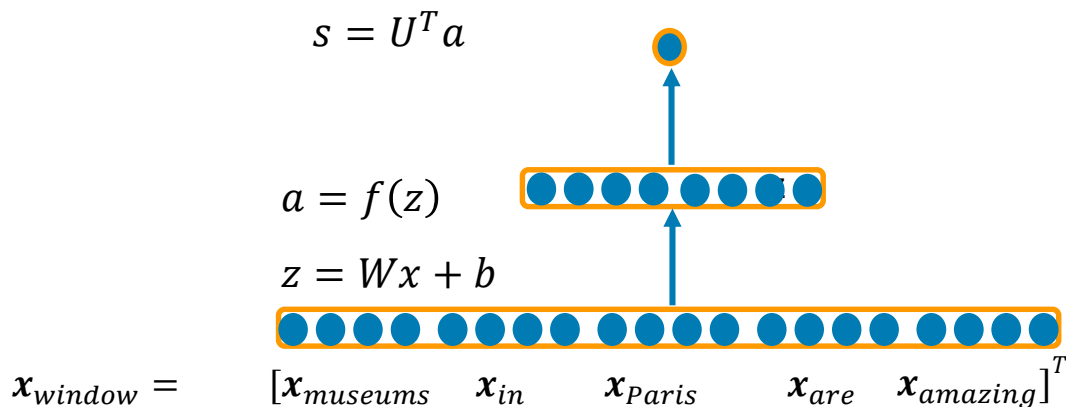


- Input: Concatenated word vectors of a window x
- Hidden layer: $z = Wx + b$ with activation function $a = f(z)$
- Output layer:
 - Softmax for probability distribution over multiple classes
 - $p(y|x) = \text{softmax}(Wa)$
 - Easier: non-normalized score for each class
 - $\text{score}(x) = U^T a \in \mathbb{R}$

Forward Pass



- Score for one window:
 - $s = \text{score}(\text{museums in Paris are amazing})$
 - $s = \mathbf{U}^T f(\mathbf{W}\mathbf{x} + \mathbf{b})$ $\mathbf{x} \in \mathbb{R}^{20 \times 1}$, $\mathbf{W} \in \mathbb{R}^{8 \times 20}$, $\mathbf{U} \in \mathbb{R}^{8 \times 1}$



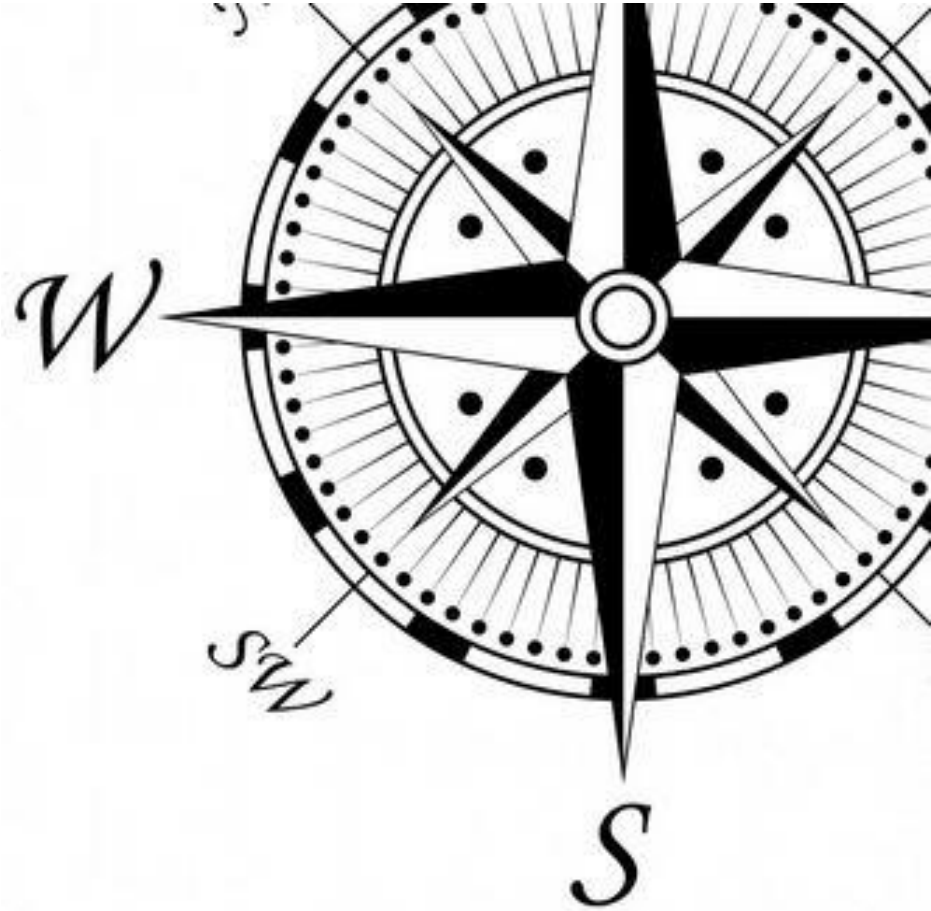


- How does the network architecture look like if softmax over four classes should be used for the output layer?
- Which effect do the window size and the dimensionality of the word vectors have?
 - On runtime,
 - memory,
 - Quality of results?




Topics Today

1. Text as Input to Neural Networks
2. DNN to Classify Words
3. **The Max-Margin Loss Function**



Max-Margin Loss Function Idea

- Many windows to train
 - Most of them don't have a location in the center
 - Those are called „corrupt“ or negative samples
- Example
 - $s = \text{score}(\text{museums in Paris are amazing})$
 - $s_c = \text{score}(\text{Not all museums in Paris})$
- Score of positive examples should be high; score of negative examples should be low
- Target function 
$$J = \max(0, 1 - s + s_c)$$
 - Not differentiable but continuous
 - SGD

For One Training Sample

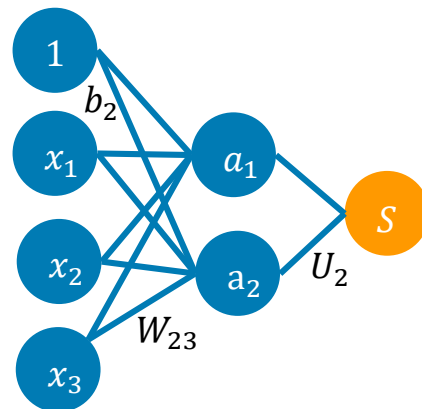


- $J = \max(0, 1 - s + s_c)$
- For each positive window, i.e. with a location in the center
 - Sample m negative windows
 - Sum over these training windows
- The score for each positive window should be at least 1.0 higher than the highest score of the negative windows

Training with Backpropagation

- $J = \max(0, 1 - s + s_c)$
 - Assumption at start $J > 0$
- $s = U^T f(Wx + b)$
- $s_c = U^T f(Wx_c + b)$
- Derivatives of s and s_c with respect to all involved variables: U, W, b, x
- Gradient with Respect to U

$$\frac{\partial s}{\partial U} = \frac{\partial}{\partial U} U^T a = a$$



Gradient with Respect to W

- $\frac{\partial s}{\partial W} = \frac{\partial}{\partial W} U^T a = \frac{\partial}{\partial W} U^T f(z) = \frac{\partial}{\partial W} U^T f(Wx + b)$
- Derivative of single weight W_{ij}

$$\frac{\partial s}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} U_i a_i$$

$$= U_i \frac{\partial}{\partial W_{ij}} a_i$$

$$= U_i \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} = U_i \frac{\partial f(z_i)}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}}$$

$$= U_i f'(z_i) \frac{\partial z_i}{\partial W_{ij}} = U_i f'(z_i) \frac{\partial W_{i \cdot} x + b_i}{\partial W_{ij}}$$

$$= U_i f'(z_i) \frac{\partial}{\partial W_{ij}} \sum_k W_{ik} x_k$$

W_{ij} only appears within a_i

W_{ij} only appears together with x_j

$$= U_i f'(z_i) x_j$$

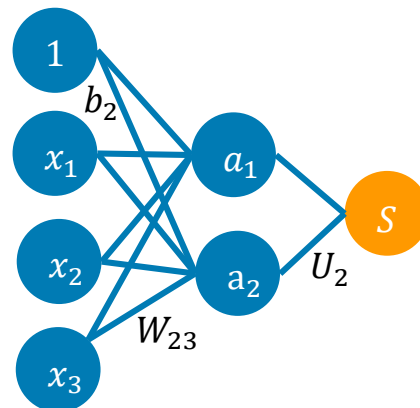
with $f'(z) = f(z)(1 - f(z))$ for the logistic function

Local error signal δ_i

Local input signal x_j

$$z_i = W_{i \cdot} x + b_i = \sum_k W_{ik} x_k + b_i$$

$$a_i = f(z_i)$$



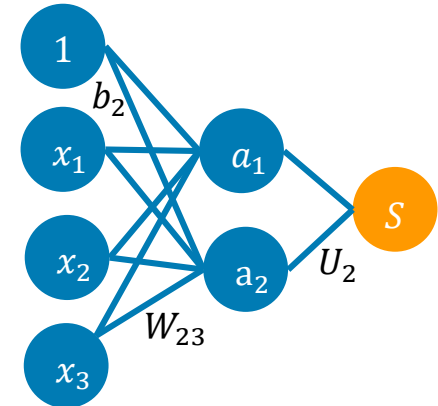
Gradient with Respect to W

- $\frac{\partial s}{\partial W} = \frac{\partial}{\partial W} U^T a = \frac{\partial}{\partial W} U^T f(z) = \frac{\partial}{\partial W} U^T f(Wx + b)$
 - From single weight W_{ij} to full W :

$$\frac{\partial s}{\partial W_{ij}} = U_i f'(z_i) x_j = \delta_i x_j$$
 - We want all combinations $i=1,2$ and $j=1,2,3$
 - We need all, i.e., complete matrix W
 - Solution:
 - Dyadic product (outer product), often denoted \otimes
 - In contrast to scalar product (inner product)
- $\frac{\partial s}{\partial W} = \delta x^T$ with $\delta \in R^{2 \times 1}$ is the error signal coming from each activation a

$$z_i = W_i \cdot x + b_i = \sum_k W_{ik} x_k + b_i$$

$$a_i = f(z_i)$$

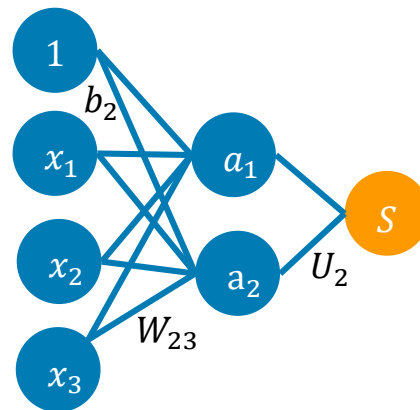


Dimensions of $\frac{\partial s}{\partial W}$?

Gradient with Respect to b



- $J = \max(0, 1 - s + s_c)$
- $s = U^T f(Wx + b)$
- $s_c = U^T f(Wx_c + b)$
- $\frac{\partial s}{\partial U} = \frac{\partial}{\partial U} U^T a = a$
- $\frac{\partial s}{\partial W} = \delta x^T$
- $\frac{\partial s}{\partial b_i} = U_i f'(z_i) \frac{\partial W_i x + b_i}{\partial b_i} = \delta_i$
- $\frac{\partial s}{\partial b} = \delta$



Gradient with Respect to x

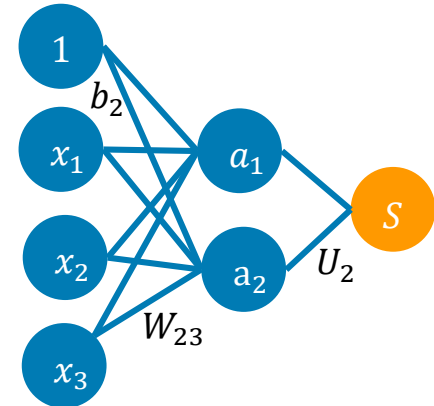
- $J = \max(0, 1 - s + s_c)$
- $s = U^T f(Wx + b)$
- $s_c = U^T f(Wx_c + b)$

- $\frac{\partial s}{\partial x_j} = \sum_{i=1}^2 \frac{\partial s}{\partial a_i} \frac{\partial a_i}{\partial x_j}$
- $= \sum_{i=1}^2 \frac{\partial U^T a}{\partial a_i} \frac{\partial a_i}{\partial x_j}$
- $= \sum_{i=1}^2 U_i \frac{\partial f(W_i \cdot x + b)}{\partial x_j}$
- $= \sum_{i=1}^2 U_i f'(z_i) \frac{\partial W_i \cdot x}{\partial x_j}$
- $= \sum_{i=1}^2 \delta_i W_{ij}$
- $= W_{.j}^T \delta$

$$\frac{\partial s}{\partial x} = W^T \delta$$

Dimensions of $\frac{\partial s}{\partial x}$?

Dimensions of $\frac{\partial s}{\partial x_j}$?



All Gradients

- $J = \max(0, 1 - s + s_c)$
- $s = U^T f(Wx + b)$
- $s_c = U^T f(Wx_c + b)$
- Derivatives of s and s_c with respect to all involved variables: U, W, b, x
- $\frac{\partial s}{\partial U} = \frac{\partial}{\partial U} U^T a = a$
- $\frac{\partial s}{\partial W} = \delta x^T$
- $\frac{\partial s}{\partial b} = \delta$
- $\frac{\partial s}{\partial x} = W^T \delta$
- From s to J : indicator function: $\mathbb{I}[1 - s + s_c > 0] = \begin{cases} 1 & \text{if } 1 - s + s_c > 0 \\ 0 & \text{else} \end{cases}$
- E.g. gradient of J with respect to U : $\frac{\partial J}{\partial U} = \mathbb{I}[1 - s + s_c > 0](-a + a_c)$

Learning Goals for this Chapter



- Know different input representations for text
 - and how/when to use them
 - Develop a simple window classifier
 - Understand the max-margin loss function
 - and know how to employ it
-
- Relevant chapters
 - P6.1
 - S3 (2019) <https://www.youtube.com/watch?v=PLryWeHPcBs>

- Li, J., Sun, A., Han, J., & Li, C. (2020). A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1), 50-70.
 - <https://arxiv.org/pdf/1812.09449.pdf>