MLPs, Representations, Ngram Language Models

Yoav Goldberg

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

Linear Classifier

Binary:

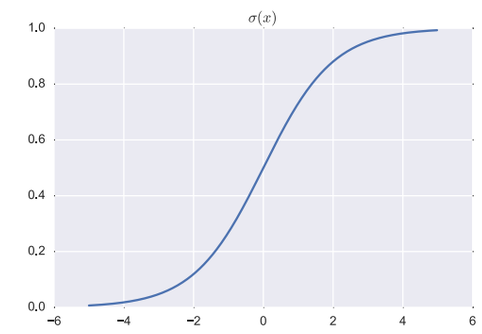
$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \qquad \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

Linear Classifier

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



Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \qquad \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

$$\arg\max_i (\mathbf{W} \cdot \mathbf{x} + \mathbf{b})_{[i]}$$

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

$$\arg\max_i (\mathbf{W} \cdot \mathbf{x} + \mathbf{b})_{[i]}$$

$$\arg\max_i \text{softmax}(\mathbf{W} \cdot \mathbf{x} + \cdot b)_{[i]}$$

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

$$\arg\max_i (\mathbf{W} \cdot \mathbf{x} + \mathbf{b})_{[i]}$$

is this really needed?

$$\arg\max_i \text{softmax}(\mathbf{W} \cdot \mathbf{x} + \cdot b)_{[i]}$$

XOR

$$\text{xor}(0,0) = 0$$

$$\text{xor}(1,0) = 1$$

$$\text{xor}(0,1) = 1$$

$$\text{xor}(1,1) = 0$$

XOR

$$(0,0) \cdot \mathbf{w} + b < 0$$

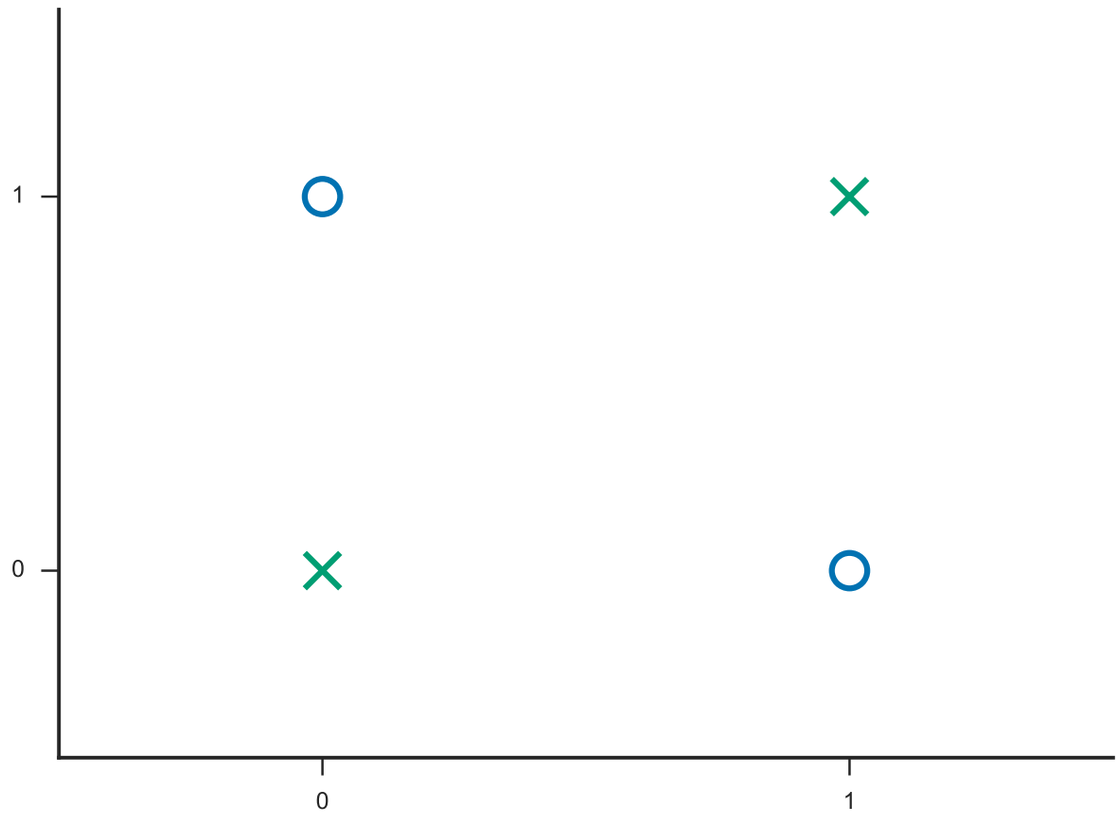
$$(0,1) \cdot \mathbf{w} + b \geq 0$$

$$(1,0) \cdot \mathbf{w} + b \geq 0$$

$$(1,1) \cdot \mathbf{w} + b < 0$$

$$\mathbf{w} = ?$$

XOR

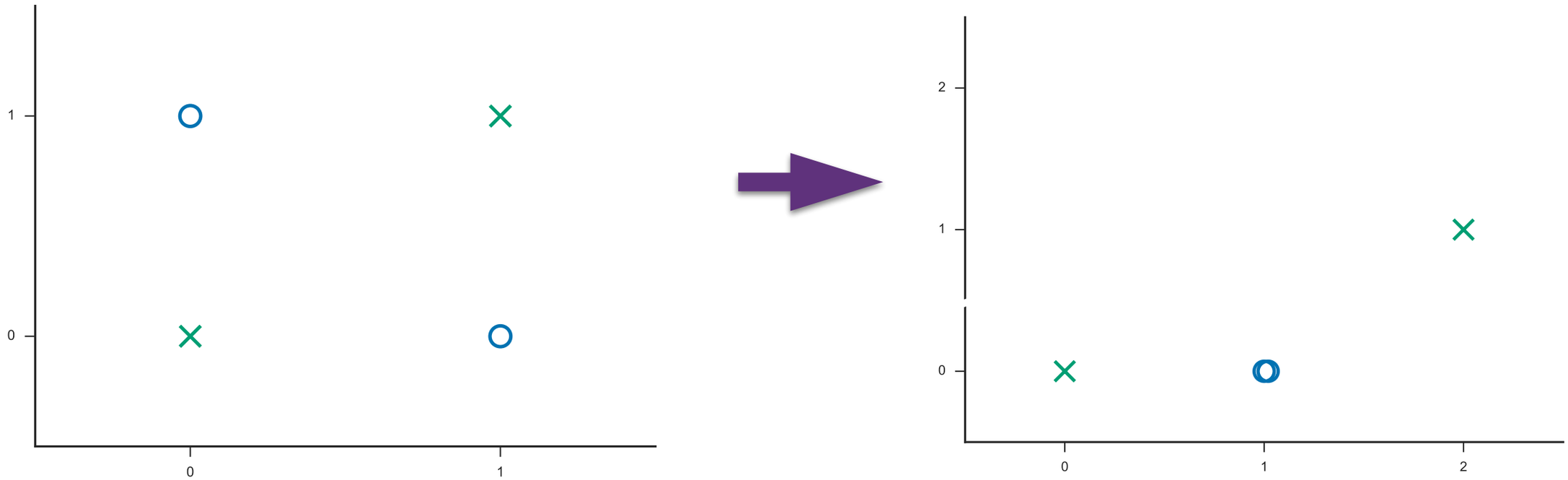


XOR

$$\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2],$$

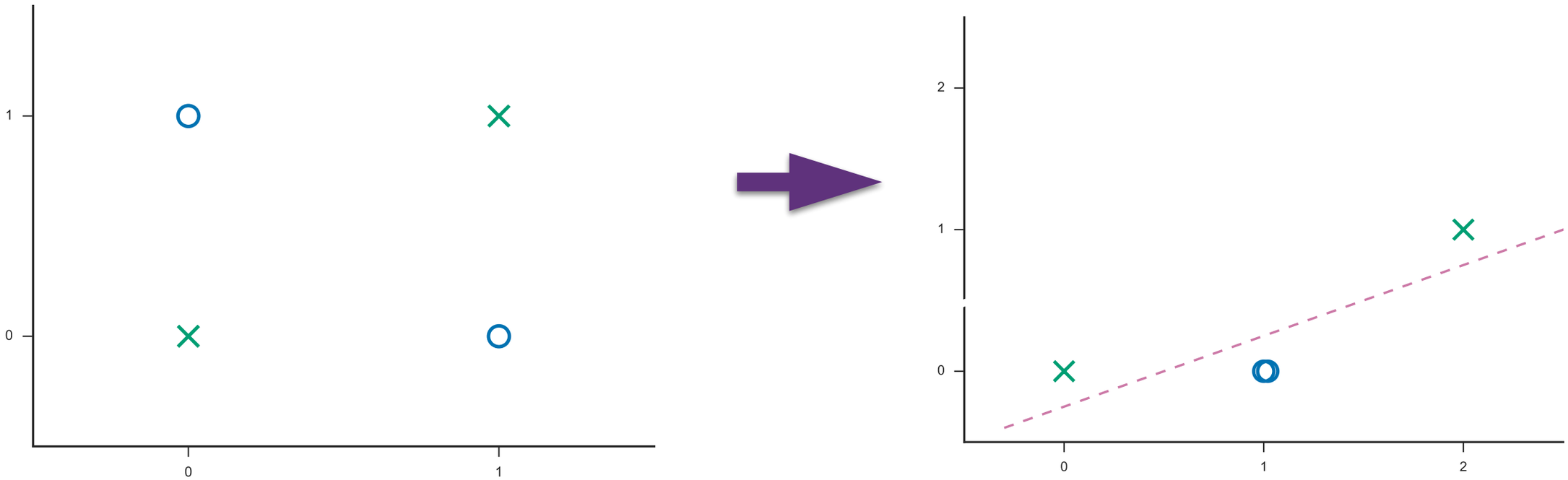
XOR

$$\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2],$$



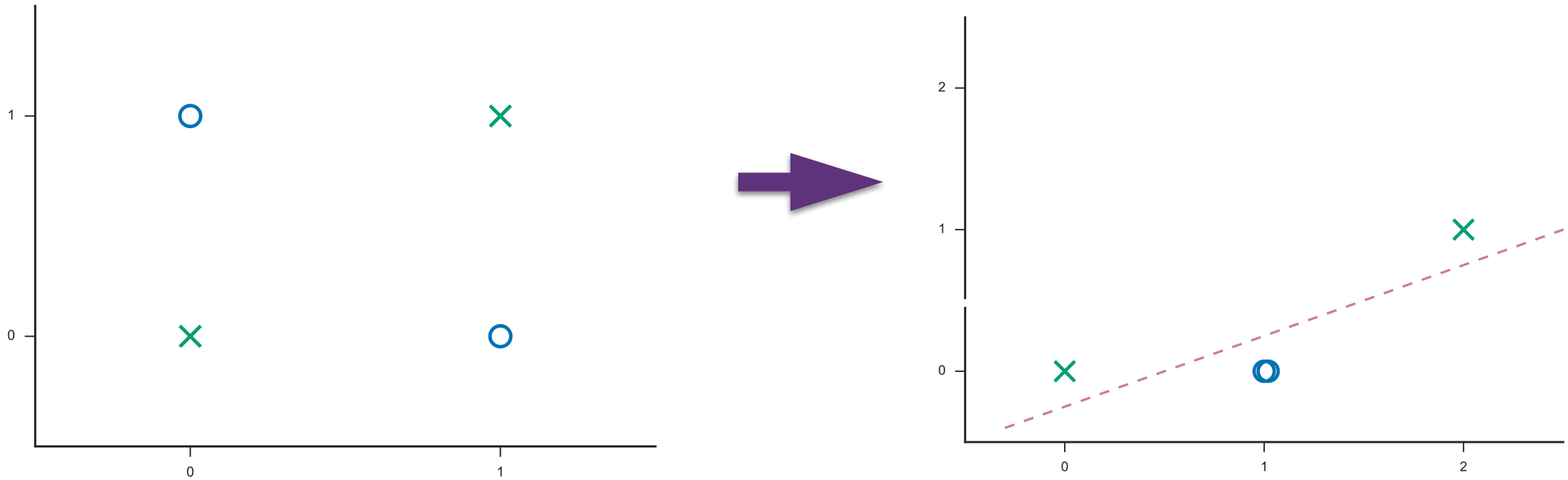
XOR

$$\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2],$$



XOR

$$\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2],$$



$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b$$

XOR

$$\phi(x_1, x_2) = [x_1 \times x_2, x_1 + x_2],$$

- Can make the transfer function **trainable**:

$$\phi(\mathbf{x}) = g(\mathbf{W}' \cdot \mathbf{x} + \mathbf{b}')$$

$$g(x) = \max(x, 0)$$

Multilayer Networks

Linear Classifier

Binary:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

$$\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$\sigma(\mathbf{w} \cdot \mathbf{x} + b)$$

Multi class:

$$f(\mathbf{x}) = \mathbf{W} \cdot \mathbf{x} + \mathbf{b}$$

$$\arg\max_i \text{softmax}(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})_{[i]}$$

Non-Linear Classifier

$$f_\theta(\mathbf{x}) \ = \mathbf{w} g(\mathbf{W}' \cdot \mathbf{x} + \mathbf{b}') + b$$

Non-Linear Classifier

Multi-layer Perceptron (MLP):

$$f_\theta(\mathbf{x}) = \ \mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) = \mathbf{y}$$

$$\mathbf{h^1} = g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})$$

$$\mathbf{h^2} = g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2})$$

$$\mathbf{y} = \mathbf{h^2}\mathbf{W^3}$$

Non-Linear Classifier

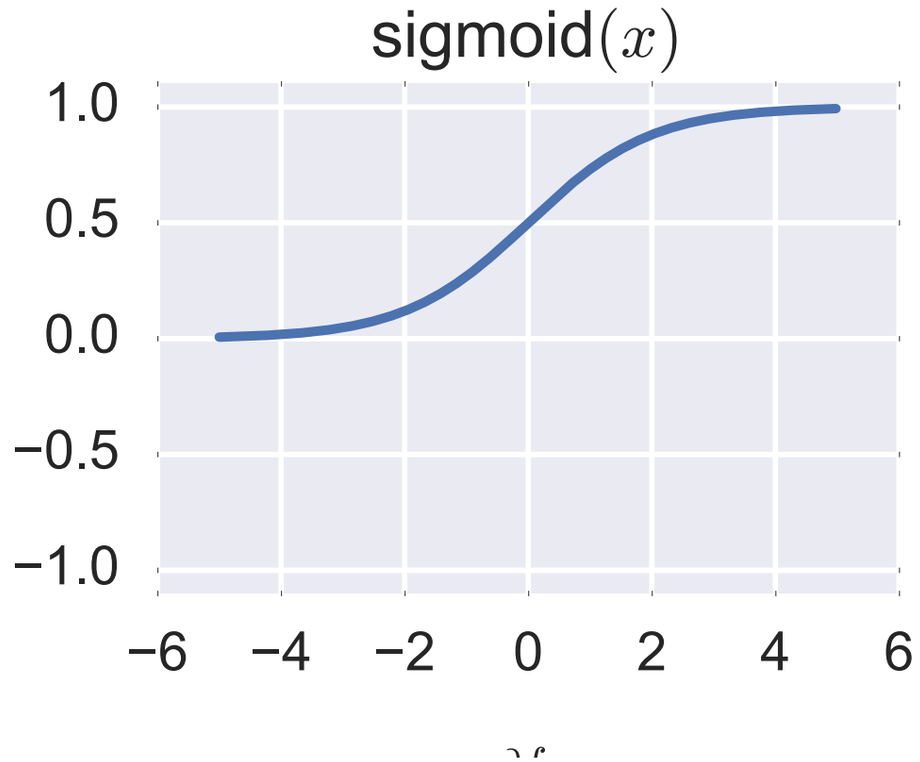
Multi-layer Perceptron (MLP):

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

Common Non-linearities

Sigmoid

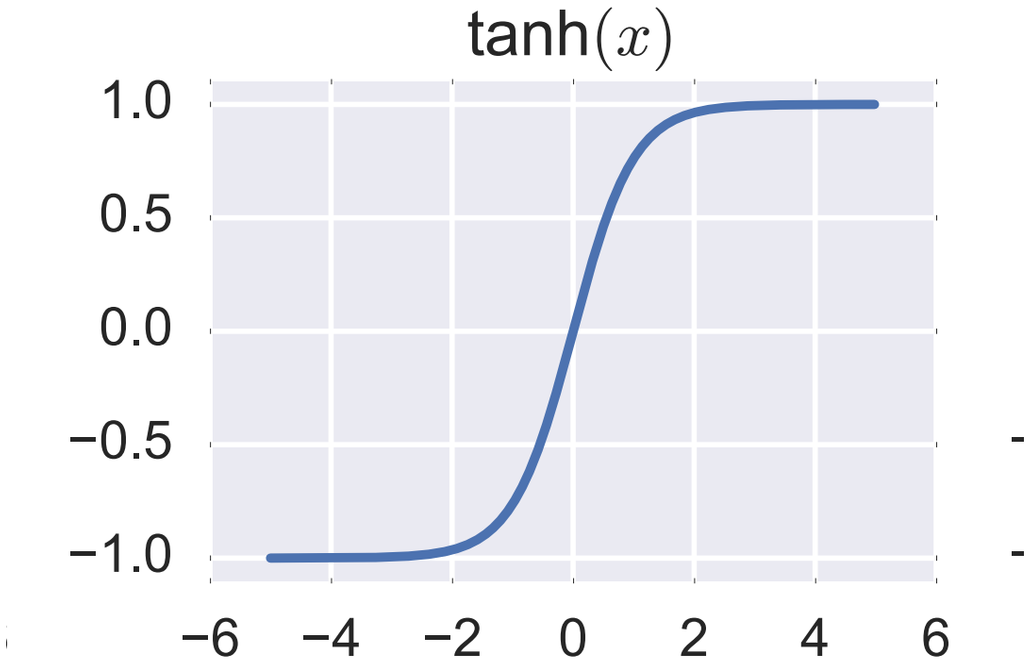
$$\sigma(x) = 1/(1 + e^{-x})$$



Common Non-linearities

tanh

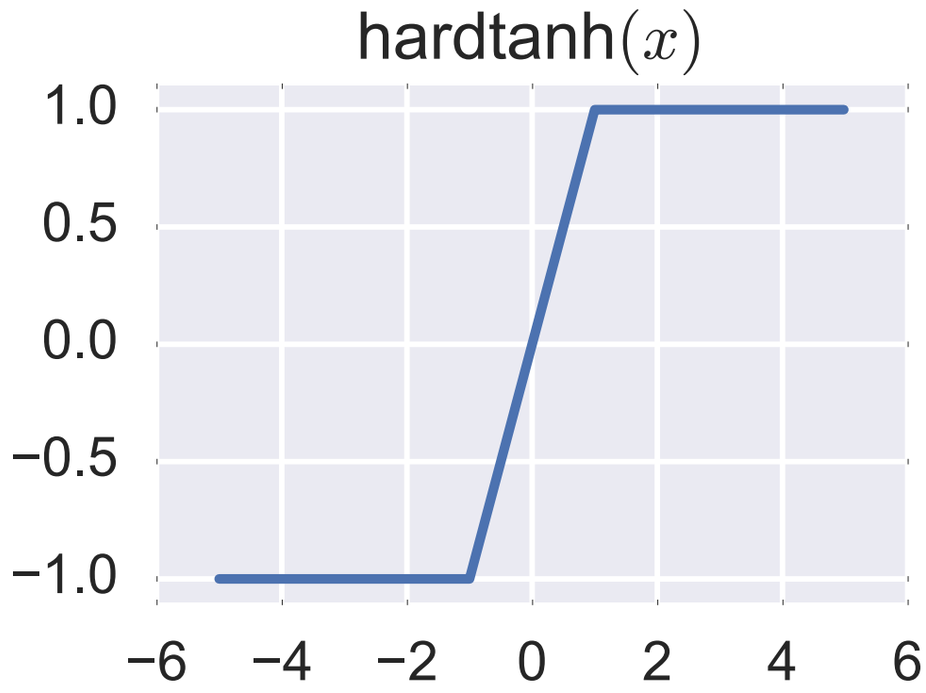
$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$$



Common Non-linearities

hard-tanh

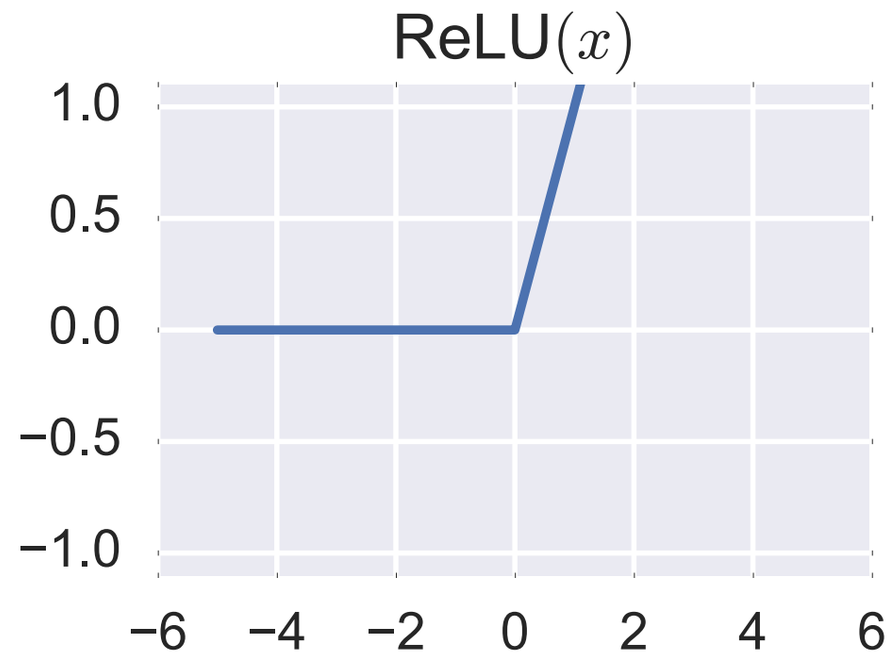
$$\text{hardtanh}(x) = \begin{cases} -1 & x < -1 \\ 1 & x > 1 \\ x & \text{otherwise} \end{cases}$$



Common Non-linearities

ReLU (rectifier, rectified linear unit)

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



How many layers to use? And how wide should they be?

- No hard and fast rules.
- In vision, we see that "deeper is better".
  - Not always the case in text / sequences (though with transformers, we may be starting to see this).
- Can think of each layer as **transforming** the previous layer (remember the xor example).
- Narrower layers "compress" the information in the previous layer. Wider layers introduce redundancies.

Dropout

- **At each iteration, select a random subset of "neurons" and "drop" them.**

- Like training 2^n different networks.

- Prevents co-adaptation of neurons (prevents neurons from depending on each other).

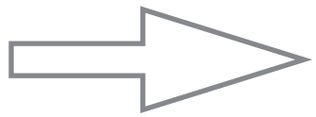
Dropout

$$\mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) = \mathbf{y}$$

$$\mathbf{h^1} = g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})$$

$$\mathbf{h^2} = g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2})$$

$$\mathbf{y} = \mathbf{h^2}\mathbf{W^3}$$



$$\mathrm{NN}_{\mathrm{MLP2}}(\mathbf{x}) = \mathbf{y}$$

$$\mathbf{h^1} = g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})$$

$$\mathbf{m^1} \sim \mathrm{Bernouli}(r^1)$$

$$\tilde{\mathbf{h}}^{\mathbf{1}} = \mathbf{m^1} \odot \mathbf{h^1}$$

$$\mathbf{h^2} = g^2(\tilde{\mathbf{h}}^{\mathbf{1}}\mathbf{W^2} + \mathbf{b^2})$$

$$\mathbf{m^2} \sim \mathrm{Bernouli}(r^2)$$

$$\tilde{\mathbf{h}}^{\mathbf{2}} = \mathbf{m^2} \odot \mathbf{h^2}$$

$$\mathbf{y} = \tilde{\mathbf{h}}^{\mathbf{2}}\mathbf{W^3}$$

Initialization

- With a (log) linear model, initialization doesn't matter much.

- With MLPs or more complex networks, initialization is **crucial** for achieving good performance.

Initialization

- With a (log) linear model, initialization doesn't matter much.
- With MLPs or more complex networks, initialization is **crucial** for achieving good performance.

$$\mathbb{R}^{n \times m} \sim \text{Uniform}[-\epsilon, +\epsilon]$$

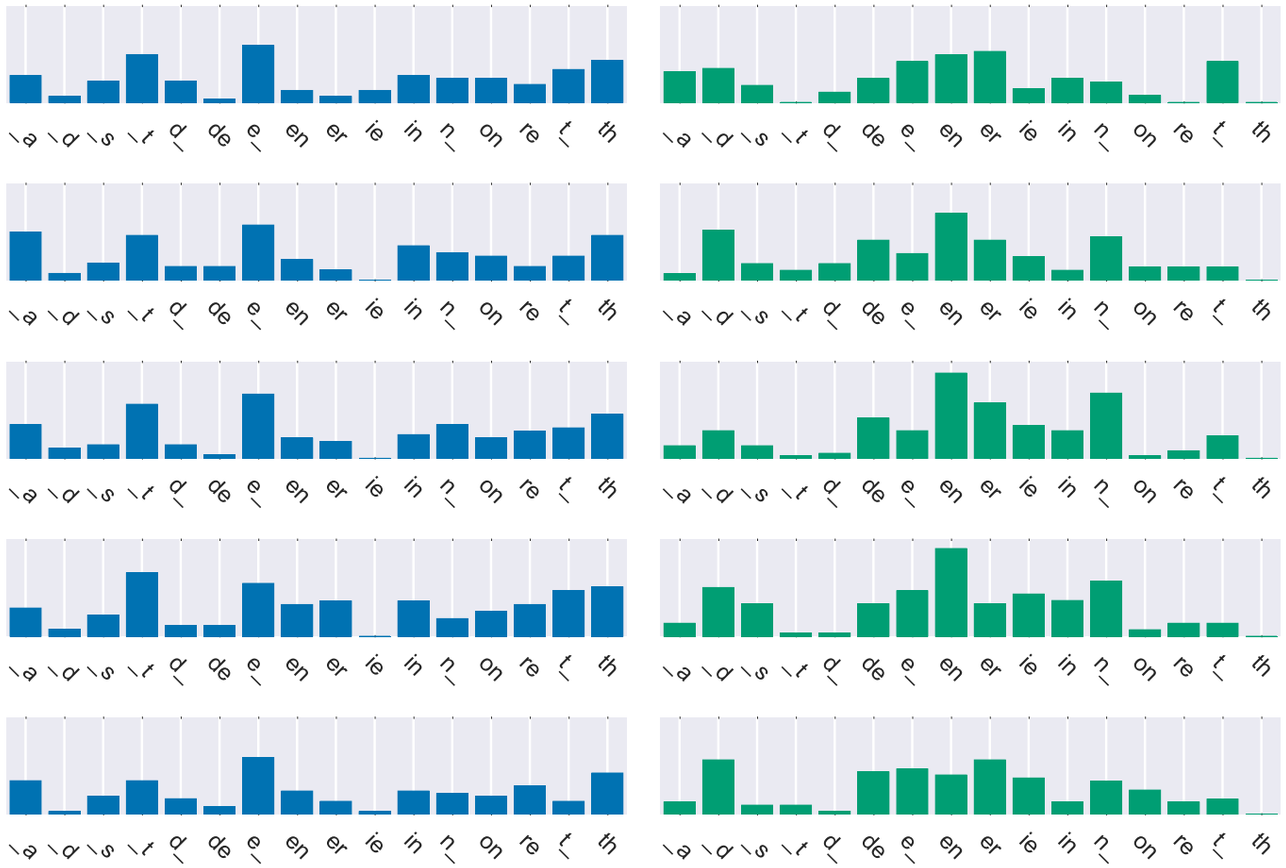
Xavier Glorot et al's suggestion:

$$\epsilon = \frac{\sqrt{(6)}}{\sqrt{m+n}}$$

Representations

Representations

Language Identification for 6 languages based on letter bigram counts.



$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

assume 28 letters (including space).

the vector **x** is 784 dimensional vector.

each entry is the count for a particular letter pair.

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

consider the values $\hat{\mathbf{y}}$

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

consider the 6 columns of **W**

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

consider the 784 rows of **W**

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

think of **x** as a sum of one-hot vectors.

what is **xW** ?

$$\hat{\mathbf{y}} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{W} + \mathbf{b}$$

$$\hat{y} = \underset{i}{\operatorname{argmax}}\, \hat{\mathbf{y}}_{[i]}$$

Representations

what happens if we add layers?

$$\hat{\mathbf{y}} = g(\mathbf{x}\mathbf{W})\mathbf{U}$$

$$\mathbf{W} \in \mathbb{R}^{784\times 30} \quad \mathbf{U} \in \mathbb{R}^{30\times 6}$$

Language Modeling

Let's talk about sequences

- Predicting how a sequence will continue.

מה אתה _

Language Model

$$p(x_i|x_1, ..., x_{i-1})$$

Language Model: Markov Assumption

$$p(x_i|x_1, ..., x_{i-1}) \approx p(x_i|x_{i-4}, x_{i-3}, x_{i-2}, x_{i-1})$$

Language Model: Markov Assumption

$$p(x_i|x_1, ..., x_{i-1}) \approx p(x_i|x_{i-4}, x_{i-3}, x_{i-2}, x_{i-1})$$

(condition only on last n items)

this is called n-gram language model

Language Model

- LM can also be used to assign a probability to a sequence.

$$
\begin{aligned}
p(x_1, ..., x_n) = & p_{LM}(x_1 | \texttt{*S*}, \texttt{*S*}) \\
& \times p_{LM}(x_2 | \texttt{*S*}, x_1) \\
& \times p_{LM}(x_3 | x_1, x_2) \\
& \times p_{LM}(x_4 | x_2, x_3) \\
& \dots \\
& \times p_{LM}(x_n | x_{n-2}, x_{n-1})
\end{aligned}
$$

Language Model

- Very useful (used in Speech Recognition, Machine Translation.. and many others).
- Does not have to be over natural language.
- Huge research topic. We'll see a neural LM.

Neural LM

$$p(x_k|x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1}) = \text{softmax}(\text{MLP}(\mathbf{x}))$$

$$\mathbf{x} = encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

Neural LM

$$p(x_k|x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1}) = \text{softmax}(\text{MLP}(\mathbf{x}))$$

$$\mathbf{x} = \mathit{encode}(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

$$\text{softmax}(g(g(\mathbf{x}\mathbf{W^1} + \mathbf{b^1})\mathbf{W^2} + \mathbf{b^2})\mathbf{W^3} + \mathbf{b^3})$$

$$\text{softmax}(\square)$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow$$

$$\mathbf{x}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

$$\mathbb{R}^{d_{out}}$$

$$\text{softmax}(\square)$$

$$\mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\mathbb{R}^{d_2}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\mathbb{R}^{d_1}$$

$$\uparrow$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\mathbb{R}^{d_{in}}$$

$$\uparrow$$

$$\mathbf{x}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

$\mathbb{R}^{d_{out}}$

$\text{softmax}(\square)$

$\mathbb{R}^{d_{out}}$

$\uparrow$

$\square\mathbf{W^3} + \mathbf{b^3}$

$\mathbb{R}^{d_2}$

$\uparrow$

$g(\square\mathbf{W^2} + \mathbf{b^2})$

$\mathbb{R}^{d_1}$

$\uparrow$

$g(\square\mathbf{W^1} + \mathbf{b^1})$

$\mathbb{R}^{d_{in}}$

$\uparrow$

$\mathbf{x}$

$\uparrow$

$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$

Encoding k elements

$$\text{encode}(x_1, x_2, x_3, x_4)$$

We have k elements in a vocabulary of size |V|

Encoding k elements

$$\text{encode}(x_1, x_2, x_3, x_4)$$

We have k elements in a vocabulary of size |V|

4

10

V={A,B,C,D,E,F,G,H,I,J}

Encoding k elements

$$\mathrm{encode}(D, A, G, C)$$

A= [1,0,0,0,0,0,0,0,0,0]
B= [0,1,0,0,0,0,0,0,0,0]
C= [0,0,1,0,0,0,0,0,0,0]
D= [0,0,0,1,0,0,0,0,0,0]
E= [0,0,0,0,1,0,0,0,0,0]
F= [0,0,0,0,0,1,0,0,0,0]
G= [0,0,0,0,0,0,1,0,0,0]
H= [0,0,0,0,0,0,0,1,0,0]
I= [0,0,0,0,0,0,0,0,1,0]
J= [0,0,0,0,0,0,0,0,0,1]

Encoding k elements

$$\text{encode}(D, A, G, C)$$

A= [1,0,0,0,0,0,0,0,0,0]
B= [0,1,0,0,0,0,0,0,0,0]
C= [0,0,1,0,0,0,0,0,0,0]
D= [0,0,0,1,0,0,0,0,0,0]
E= [0,0,0,0,1,0,0,0,0,0]
F= [0,0,0,0,0,1,0,0,0,0]
G= [0,0,0,0,0,0,1,0,0,0]
H= [0,0,0,0,0,0,0,1,0,0]
I= [0,0,0,0,0,0,0,0,1,0]
J= [0,0,0,0,0,0,0,0,0,1]

$$\mathbf{v}_D + \mathbf{v}_A + \mathbf{v}_G + \mathbf{v}_C$$

[0,0,0,1,0,0,0,0,0,0]
+
[1,0,0,0,0,0,0,0,0,0]
+
[0,0,0,0,0,0,1,0,0,0]
+
[0,0,1,0,0,0,0,0,0,0]
=
[1,0,0,1,0,0,1,0,0,0]

Encoding k elements

$$\text{encode}(D, A, G, C)$$

A= [1,0,0,0,0,0,0,0,0,0]
B= [0,1,0,0,0,0,0,0,0,0]
C= [0,0,1,0,0,0,0,0,0,0]
D= [0,0,0,1,0,0,0,0,0,0]
E= [0,0,0,0,1,0,0,0,0,0]
F= [0,0,0,0,0,1,0,0,0,0]
G= [0,0,0,0,0,0,1,0,0,0]
H= [0,0,0,0,0,0,0,1,0,0]
I= [0,0,0,0,0,0,0,0,1,0]
J= [0,0,0,0,0,0,0,0,0,1]

$$\mathbf{v}_D + \mathbf{v}_A + \mathbf{v}_G + \mathbf{v}_C$$

[0,0,0,1,0,0,0,0,0,0]
+
[1,0,0,0,0,0,0,0,0,0]
+
[0,0,0,0,0,0,1,0,0,0]
+
[0,0,1,0,0,0,0,0,0,0]
=
[1,0,1,1,0,0,1,0,0,0]

what does this miss?

Encoding k elements

$$\text{encode}(D, A, G, C)$$

A= [1,0,0,0,0,0,0,0,0,0]
B= [0,1,0,0,0,0,0,0,0,0]
C= [0,0,1,0,0,0,0,0,0,0]
D= [0,0,0,1,0,0,0,0,0,0]
E= [0,0,0,0,1,0,0,0,0,0]
F= [0,0,0,0,0,1,0,0,0,0]
G= [0,0,0,0,0,0,1,0,0,0]
H= [0,0,0,0,0,0,0,1,0,0]
I= [0,0,0,0,0,0,0,0,1,0]
J= [0,0,0,0,0,0,0,0,0,1]

$$\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C$$

Encoding k elements

$$\text{encode}(D, A, G, C)$$

$$\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C$$

[0,0,0,1,0,0,0,0,0,0]
o
[1,0,0,0,0,0,0,0,0,0]
o
[0,0,0,0,0,0,1,0,0,0]
o
[0,0,1,0,0,0,0,0,0,0]
=
[0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

$\mathbb{R}^{d_{out}}$

$\text{softmax}(\square)$

$\mathbb{R}^{d_{out}}$

$\uparrow$

$\square\mathbf{W^3} + \mathbf{b^3}$

$\mathbb{R}^{d_2}$

$\uparrow$

$g(\square\mathbf{W^2} + \mathbf{b^2})$

$\mathbb{R}^{d_1}$

$\uparrow$

$g(\square\mathbf{W^1} + \mathbf{b^1})$

$\mathbb{R}^{d_{in}}$

$\uparrow$

$\mathbf{x}$

$\uparrow$

$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$

$$(\mathbf{v}_D + \mathbf{v}_A + \mathbf{v}_G + \mathbf{v}_C)\mathbf{W}$$

[0,0,0,1,0,0,0,0,0,0]
+
[1,0,0,0,0,0,0,0,0,0]
+
[0,0,0,0,0,0,1,0,0,0]
+
[0,0,1,0,0,0,0,0,0,0]
=
[1,0,0,1,0,0,1,0,0,0]

W

A= [-0.32, 0.09, 0.33,-0.44]
B= [0.29, 0.02,-0.46,-0.39]
C= [-0.46, 0.24,-0.16, 0.08]
D= [-0.15,-0.31, 0.34, 0.00]
E= [-0.10,-0.37, 0.01, 0.40]
F= [-0.28,-0.26,-0.24, 0.31]
G= [-0.32,-0.42,-0.21, 0.18]
H= [-0.09,-0.01, 0.06, 0.14]
I= [0.28,-0.02,-0.39, 0.12]
J= [0.23,-0.22,-0.14, 0.28]

$$(\mathbf{v}_D + \mathbf{v}_A + \mathbf{v}_G + \mathbf{v}_C)\mathbf{W}$$

$$= \mathbf{v}_D \cdot \mathbf{W} + \mathbf{v}_A \cdot \mathbf{W} + \mathbf{v}_G \cdot \mathbf{W} + \mathbf{v}_C \cdot \mathbf{W}$$

[0,0,0,1,0,0,0,0,0,0]
+
[1,0,0,0,0,0,0,0,0,0]
+
[0,0,0,0,0,0,1,0,0,0]
+
[0,0,1,0,0,0,0,0,0,0]
=
[1,0,0,1,0,0,1,0,0,0]

W

A= [-0.32, 0.09, 0.33,-0.44]
B= [0.29, 0.02,-0.46,-0.39]
C= [-0.46, 0.24,-0.16, 0.08]
D= [-0.15,-0.31, 0.34, 0.00]
E= [-0.10,-0.37, 0.01, 0.40]
F= [-0.28,-0.26,-0.24, 0.31]
G= [-0.32,-0.42,-0.21, 0.18]
H= [-0.09,-0.01, 0.06, 0.14]
I= [0.28,-0.02,-0.39, 0.12]
J= [0.23,-0.22,-0.14, 0.28]

$$(\mathbf{v}_D + \mathbf{v}_A + \mathbf{v}_G + \mathbf{v}_C)\mathbf{W}$$

$$= \mathbf{v}_D \cdot \mathbf{W} + \mathbf{v}_A \cdot \mathbf{W} + \mathbf{v}_G \cdot \mathbf{W} + \mathbf{v}_C \cdot \mathbf{W}$$

sum of rows in **W**

each row corresponds to a certain vocabulary item.

$$(\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C)\mathbf{W}$$

$$(\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C)\mathbf{W}$$

still sum of rows in **W**
but **W** has 4x many rows.

[0,0,0,1,0,0,0,0,0,0]

o

[1,0,0,0,0,0,0,0,0,0]

o

[0,0,0,0,0,0,1,0,0,0]

o

[0,0,1,0,0,0,0,0,0,0]

=

[0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0]

A(-3)= [0.42,-0.15, 0.12, 0.02]
B(-3)= [0.28,-0.15,-0.11, 0.32]
C(-3)= [0.15,-0.24, 0.23, 0.41]
D(-3)= [-0.12,-0.24, 0.12,-0.34]
E(-3)= [-0.42,-0.21, 0.08, 0.40]
F(-3)= [0.20, 0.11,-0.31, 0.33]
G(-3)= [0.07,-0.05, 0.16, 0.23]
H(-3)= [0.28, 0.03, 0.22,-0.49]
I(-3)= [0.08, 0.39,-0.25, 0.27]
J(-3)= [0.10,-0.42,-0.37, 0.35]
A(-2)= [-0.00, 0.41, 0.19, 0.49]
B(-2)= [0.24, 0.48, 0.34,-0.42]
C(-2)= [-0.46, 0.22, 0.24,-0.21]
D(-2)= [-0.11,-0.48, 0.18,-0.22]
E(-2)= [-0.32, 0.10,-0.41,-0.43]
F(-2)= [0.32, 0.02,-0.22, 0.06]
G(-2)= [-0.31,-0.36, 0.09, 0.39]
H(-2)= [0.01,-0.22,-0.09,-0.15]
I(-2)= [0.01, 0.10,-0.16,-0.21]
J(-2)= [-0.24, 0.40,-0.34,-0.13]
A(-1)= [-0.23,-0.38, 0.02, 0.32]
B(-1)= [-0.34, 0.04,-0.18,-0.00]
C(-1)= [0.40,-0.02, 0.10,-0.16]
D(-1)= [0.13,-0.07,-0.19,-0.01]
E(-1)= [0.40, 0.27,-0.33, 0.36]
F(-1)= [0.04,-0.13,-0.43, 0.39]
G(-1)= [0.44, 0.38, 0.03,-0.39]
H(-1)= [0.41,-0.23, 0.33,-0.08]
I(-1)= [-0.50,-0.16,-0.42,-0.27]
J(-1)= [-0.15, 0.41, 0.46,-0.16]
A(+0)= [-0.11, 0.03, 0.20, 0.50]
B(+0)= [0.16,-0.34, 0.20,-0.21]
C(+0)= [0.05,-0.13,-0.23,-0.31]
D(+0)= [0.13,-0.02, 0.38,-0.09]
E(+0)= [0.30, 0.39, 0.10, 0.38]
F(+0)= [-0.16,-0.31,-0.02,-0.34]
G(+0)= [0.06,-0.04, 0.02,-0.32]
H(+0)= [0.25, 0.30, 0.29, 0.24]
I(+0)= [0.40,-0.17,-0.18,-0.19]
J(+0)= [0.27, 0.33,-0.42,-0.07]

$$(\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C)\mathbf{W}$$

still sum of rows in **W**
but **W** has 4x many rows.

alternatively:

$$= \mathbf{v}_D \cdot \mathbf{W}' + \mathbf{v}_A \cdot \mathbf{W}'' + \mathbf{v}_G \cdot \mathbf{W}''' + \mathbf{v}_C \cdot \mathbf{W}''''$$

$$(\mathbf{v}_D \circ \mathbf{v}_A \circ \mathbf{v}_G \circ \mathbf{v}_C)\mathbf{W}$$

still sum of rows in **W**
but **W** has 4x many rows.

alternatively:

$$= \mathbf{v}_D \cdot \mathbf{W}' + \mathbf{v}_A \cdot \mathbf{W}'' + \mathbf{v}_G \cdot \mathbf{W}''' + \mathbf{v}_C \cdot \mathbf{W}''''$$

$$\mathbf{W} = \mathbf{W}' \circ \mathbf{W}'' \circ \mathbf{W}''' \circ \mathbf{W}''''$$

[0,0,0,1,0,0,0,0,0,0]

o

[1,0,0,0,0,0,0,0,0,0]

o

[0,0,0,0,0,0,1,0,0,0]

o

[0,0,1,0,0,0,0,0,0,0]

A(-3)= [0.42,-0.15, 0.12, 0.02]
B(-3)= [0.28,-0.15,-0.11, 0.32]
C(-3)= [0.15,-0.24, 0.23, 0.41]
D(-3)= [-0.12,-0.24, 0.12,-0.34]
E(-3)= [-0.42,-0.21, 0.08, 0.40]
F(-3)= [0.20, 0.11,-0.31, 0.33]
G(-3)= [0.07,-0.05, 0.16, 0.23]
H(-3)= [0.28, 0.03, 0.22,-0.49]
I(-3)= [0.08, 0.39,-0.25, 0.27]
J(-3)= [0.10,-0.42,-0.37, 0.35]

W'

A(-2)= [-0.00, 0.41, 0.19, 0.49]
B(-2)= [0.24, 0.48, 0.34,-0.42]
C(-2)= [-0.46, 0.22, 0.24,-0.21]
D(-2)= [-0.11,-0.48, 0.18,-0.22]
E(-2)= [-0.32, 0.10,-0.41,-0.43]
F(-2)= [0.32, 0.02,-0.22, 0.06]
G(-2)= [-0.31,-0.36, 0.09, 0.39]
H(-2)= [0.01,-0.22,-0.09,-0.15]
I(-2)= [0.01, 0.10,-0.16,-0.21]
J(-2)= [-0.24, 0.40,-0.34,-0.13]

W''

A(-1)= [-0.23,-0.38, 0.02, 0.32]
B(-1)= [-0.34, 0.04,-0.18,-0.00]
C(-1)= [0.40,-0.02, 0.10,-0.16]
D(-1)= [0.13,-0.07,-0.19,-0.01]
E(-1)= [0.40, 0.27,-0.33, 0.36]
F(-1)= [0.04,-0.13,-0.43, 0.39]
G(-1)= [0.44, 0.38, 0.03,-0.39]
H(-1)= [0.41,-0.23, 0.33,-0.08]
I(-1)= [-0.50,-0.16,-0.42,-0.27]
J(-1)= [-0.15, 0.41, 0.46,-0.16]

W'''

A(+0)= [-0.11, 0.03, 0.20, 0.50]
B(+0)= [0.16,-0.34, 0.20,-0.21]
C(+0)= [0.05,-0.13,-0.23,-0.31]
D(+0)= [0.13,-0.02, 0.38,-0.09]
E(+0)= [0.30, 0.39, 0.10, 0.38]
F(+0)= [-0.16,-0.31,-0.02,-0.34]
G(+0)= [0.06,-0.04, 0.02,-0.32]
H(+0)= [0.25, 0.30, 0.29, 0.24]
I(+0)= [0.40,-0.17,-0.18,-0.19]
J(+0)= [0.27, 0.33,-0.42,-0.07]

W''''

- 1-hot times matrix: row selection
- sum of 1-hot times matrix: row selection + sum
- concat of 1-hot: like using 1-hot from larger vocab

"Embedding Layer"

- Very common in neural network land:
  - associate each vocabulary item with a row in matrix **E** of dense vectors (dim of row << |V|)
  - concat or sum rows of **E** for input.

"Embedding Layer"

$$\text{encode}(D, A, G, C)$$

$$= \mathbf{E}_{[D]} \circ \mathbf{E}_{[A]} \circ \mathbf{E}_{[G]} \circ \mathbf{E}_{[C]}$$

E

A= [-0.32, 0.09, 0.33,-0.44]
B= [0.29, 0.02,-0.46,-0.39]
C= [-0.46, 0.24,-0.16, 0.08]
D= [-0.15,-0.31, 0.34, 0.00]
E= [-0.10,-0.37, 0.01, 0.40]
F= [-0.28,-0.26,-0.24, 0.31]
G= [-0.32,-0.42,-0.21, 0.18]
H= [-0.09,-0.01, 0.06, 0.14]
I= [0.28,-0.02,-0.39, 0.12]
J= [0.23,-0.22,-0.14, 0.28]

[-0.15,-0.31, 0.34, 0.00,-0.32, 0.09, 0.33,-0.44,-0.32,-0.42,-0.21, 0.18,-0.46, 0.24,-0.16, 0.08]

$$\mathbb{R}^{d_{out}}$$

$$\text{softmax}(\square)$$

$$\mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\mathbb{R}^{d_2}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow \qquad \mathbb{R}^{d_1}$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow \qquad \mathbb{R}^{d_{in}}$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

"Embedding Layer"

$$\text{encode}(D, A, G, C)$$

$$= \mathbf{E}_{[D]} \circ \mathbf{E}_{[A]} \circ \mathbf{E}_{[G]} \circ \mathbf{E}_{[C]}$$

E

A= [-0.32, 0.09, 0.33,-0.44]
B= [0.29, 0.02,-0.46,-0.39]
C= [-0.46, 0.24,-0.16, 0.08]
D= [-0.15,-0.31, 0.34, 0.00]
E= [-0.10,-0.37, 0.01, 0.40]
F= [-0.28,-0.26,-0.24, 0.31]
G= [-0.32,-0.42,-0.21, 0.18]
H= [-0.09,-0.01, 0.06, 0.14]
I= [0.28,-0.02,-0.39, 0.12]
J= [0.23,-0.22,-0.14, 0.28]

[-0.15,-0.31, 0.34, 0.00,-0.32, 0.09, 0.33,-0.44,-0.32,-0.42,-0.21, 0.18,-0.46, 0.24,-0.16, 0.08]

how does this relate to what we had before?

what is

$$(\mathbf{E}_{[D]} \circ \mathbf{E}_{[A]} \circ \mathbf{E}_{[G]} \circ \mathbf{E}_{[C]})\mathbf{W}$$

?

"Embedding Layer"

$$(\mathbf{E}_{[D]} \circ \mathbf{E}_{[A]} \circ \mathbf{E}_{[G]} \circ \mathbf{E}_{[C]})\mathbf{W}$$

same row in **E** regardless of position in the sequence.

but **W** will transform this row differently for each position.

$$\mathbb{R}^{d_{out}}$$

$$\text{softmax}(\square)$$

$$\mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\mathbb{R}^{d_2}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow \qquad \mathbb{R}^{d_1}$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow \qquad \mathbb{R}^{d_{in}}$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

what's in **W1**?
what's in **W2**?
what's in **W3**?

$$\mathbb{R}^{d_{out}}$$

$$\text{softmax}(\square)$$

$$\mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\mathbb{R}^{d_2}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow \qquad \mathbb{R}^{d_1}$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow \qquad \mathbb{R}^{d_{in}}$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

Neural LM

$$\text{softmax}(\square)$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

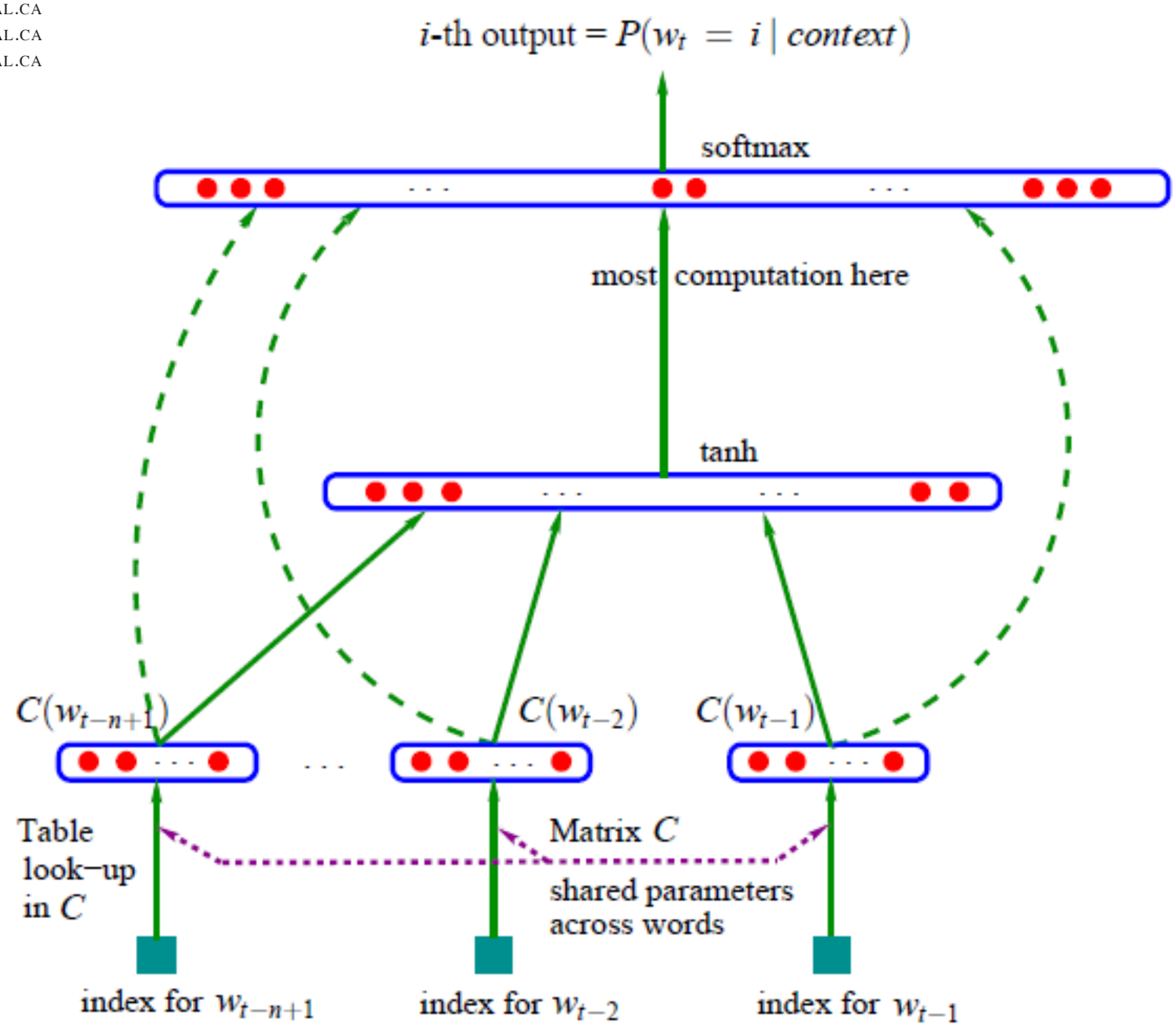
Neural LM

A Neural Probabilistic Language Model

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal, Montréal, Québec, Canada

Bengio et al 2003

Pretty much same model,
But fewer layers.



Neural LM

- What is the cost of increasing the history length?

Neural LM

- We can use a trained LM for scoring a given sentence. (how?)

Neural LM

- We can use a trained LM for comparing two given sentences. (how?)

Neural LM

- We can use a trained LM for **generating new sentences**. (how?)

Neural LM

- We can use K trained LMs for **k-class classification**. (how?)

Neural LM

- We can use K trained LMs for **k-class classification**. (how?)

$$p(w_1, w_1, ..., w_n|LM_1)$$
$$p(w_1, w_1, ..., w_n|LM_2)$$
$$p(w_1, w_1, ..., w_n|LM_3)$$

$$\hat{y} = \arg\max_k p(w_1, w_1, ..., w_n|LM_k)$$

sequence prediction tasks

- In LM:
  - predict **word i** based on **k previous words**.
- But we could also predict **label** based on **k items**.
  - which tasks can this be used for?

Neural LM

$$\text{softmax}(\square)$$

softmax over words

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

window of prev words

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

Sequence Prediction

$$\text{softmax}(\square)$$

softmax over k classes

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2})$$

$$\uparrow$$

$$g(\square\mathbf{W^1} + \mathbf{b^1})$$

$$\uparrow$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]}$$

$$\uparrow$$

sequence items

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$

classification

ACGCGCTCGATCG	V
GTGCGCTCGAACG	V
ACGCGTTCTACCG	X

tagging

DET ADJ NOUN VERB PREP DET ADJ NOUN

The brown fox jumped over the lazy dog

tagging

Window-based approach

NOUN

The brown **fox** jumped over

VERB

brown fox **jumped** over the

PREP

fox jumped **over** the lazy

back to LM

Training a language model

- Set dimensions of **E, W3**, according to vocab size.
- Initial random values for **E, W1, W2, W3, b1, b2, b3**
- For every n-tuple in some text:
    - try to predict last item based on prev n-1
    - use cross-entropy loss.

What happens after training?

- Consider the columns of **W3**.

- Consider the rows of **E**.

what's in **W3**?

columns of W3 correspond to vocab items!

$$\text{softmax}(\square) \qquad \mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$\square\mathbf{W^3} + \mathbf{b^3} \qquad \mathbb{R}^{d_{out}}$$

$$\uparrow$$

$$g(\square\mathbf{W^2} + \mathbf{b^2}) \qquad \mathbb{R}^{d_2}$$

$$\uparrow$$

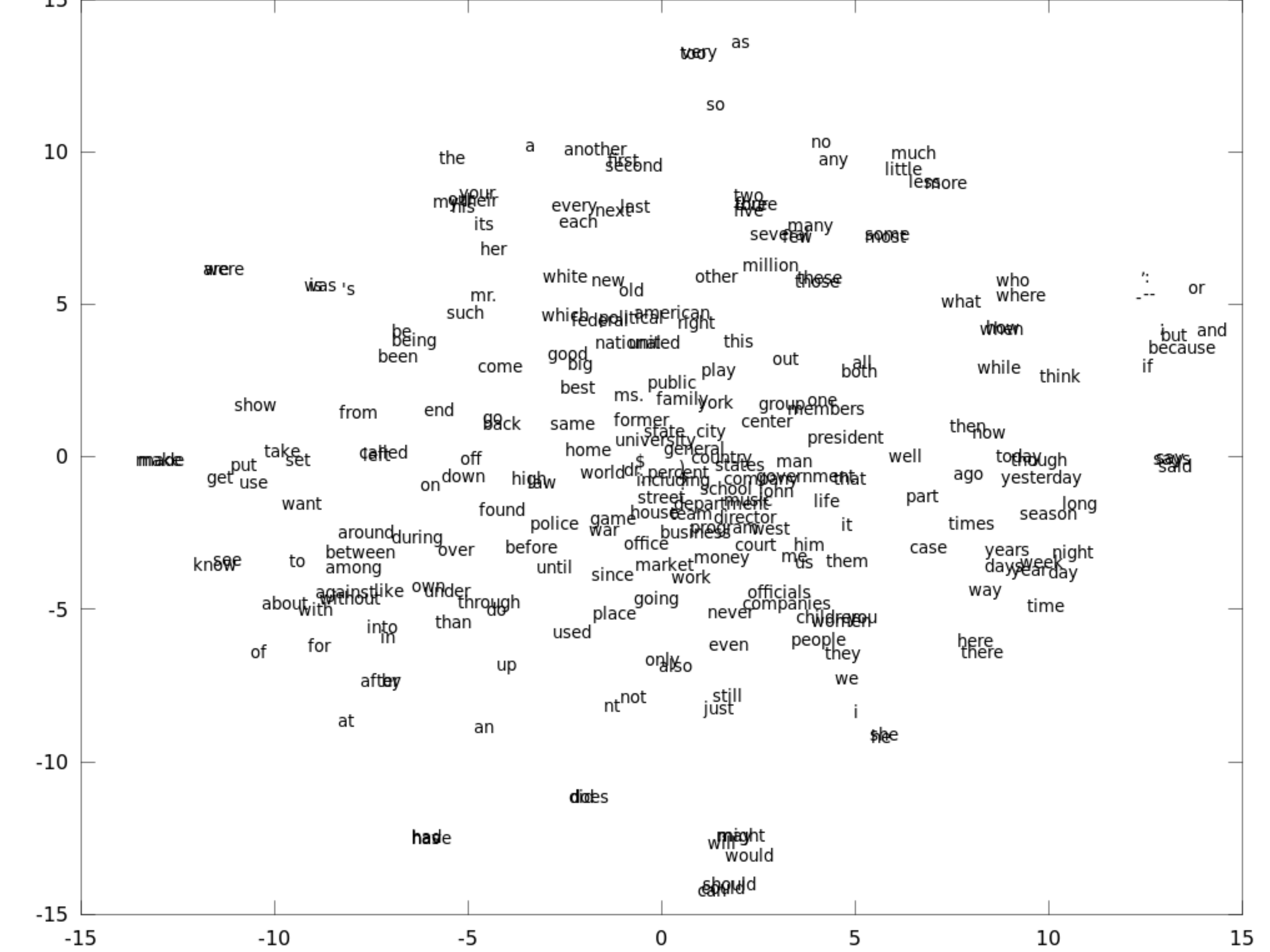
$$g(\square\mathbf{W^1} + \mathbf{b^1}) \qquad \mathbb{R}^{d_1}$$

$$\uparrow$$

$$\mathbf{E}_{[x_{k-4}]} \circ \mathbf{E}_{[x_{k-3}]} \circ \mathbf{E}_{[x_{k-2}]} \circ \mathbf{E}_{[x_{k-1}]} \qquad \mathbb{R}^{d_{in}}$$

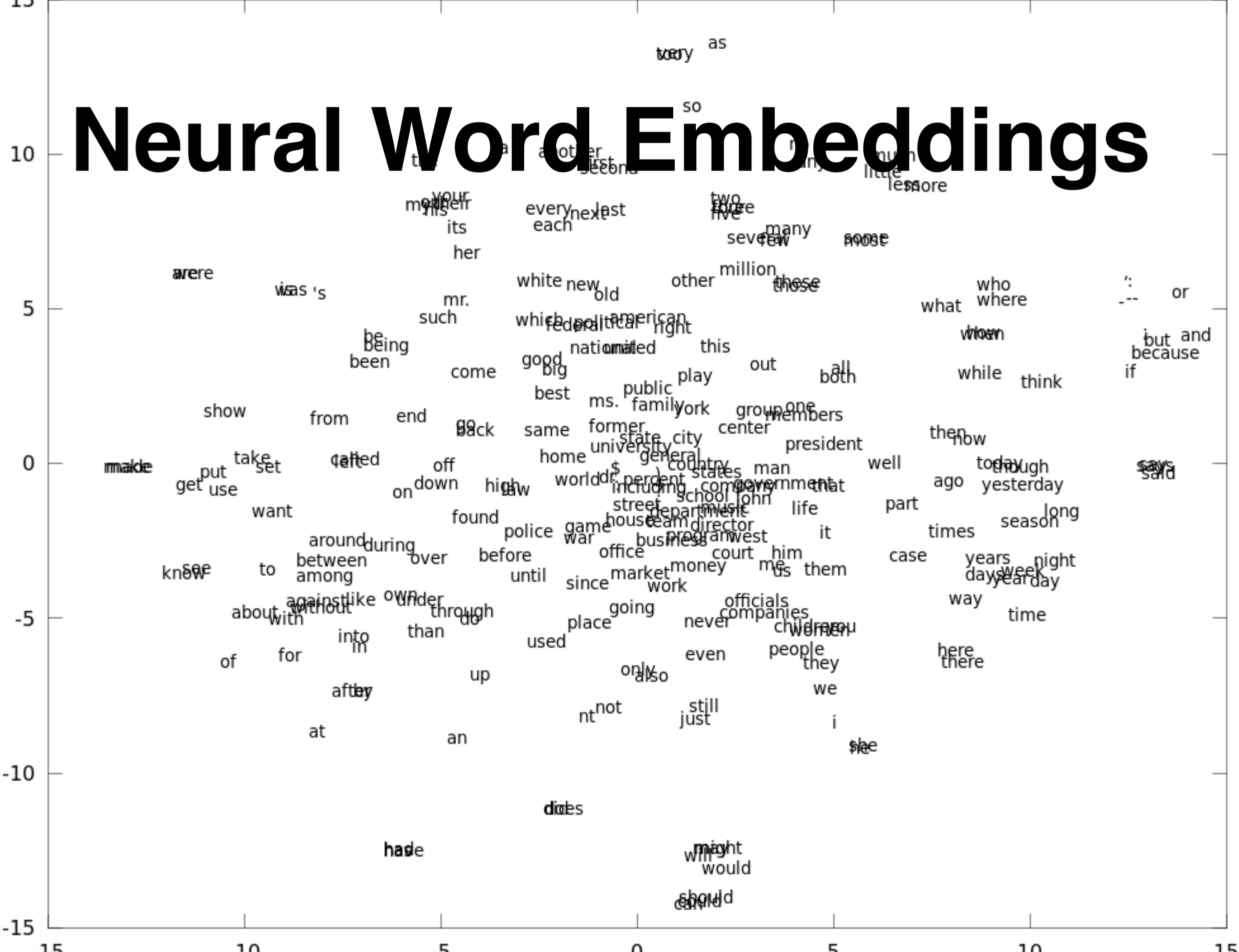
$$\uparrow$$

$$encode(x_{k-4}, x_{k-3}, x_{k-2}, x_{k-1})$$



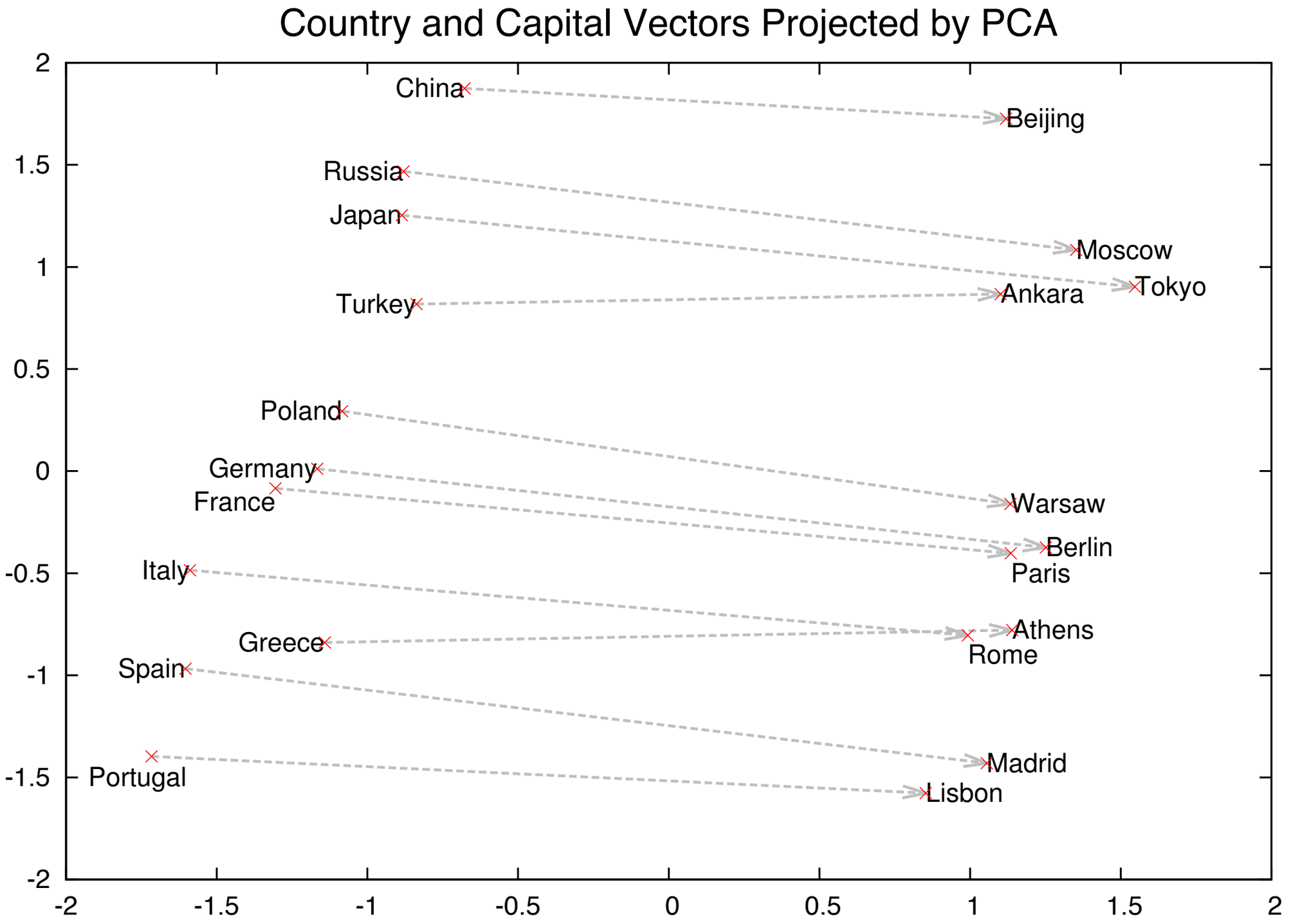
15
10
5
0
-5
-10
-15
-15
-10
-5
0
5
10
15
as
so
no
any
much
little
more
a
the
another
second
your
his
every
next
last
each
its
her
two
five
many
several
few
some
most
million
these
those
other
white
new
old
what
who
where
or
and
but
because
if
were
was
's
mr.
such
american
political
federal
right
national
united
this
be
being
been
come
good
big
out
all
both
while
think
play
public
best
ms.
family
york
group
one
members
show
from
end
back
same
former
state
city
center
president
then
now
university
general
take
called
left
home
country
states
man
well
today
though
says
said
make
made
put
set
off
world
percent
government
that
ago
yesterday
get
use
down
high
law
including
school
john
part
on
street
department
life
long
want
found
game
team
director
season
times
police
house
war
business
program
west
it
around
during
office
court
him
case
years
night
know
see
to
between
over
before
until
market
money
me
us
them
week
days
year
day
among
since
work
officials
way
about
with
without
against
like
own
under
through
going
companies
time
do
place
never
children
women
into
in
than
used
people
here
there
of
for
even
they
up
only
also
we
after
by
not
nt
still
just
i
at
an
she
he
does
did
has
have
might
may
will
would
should
could
can

Neural Word Embeddings



Neural Word Embeddings

Neural Word Embeddings



Neural Word Embeddings

Target Word	BoW5
batman	nightwing aquaman catwoman superman manhunter
hogwarts	dumbledore hallows half-blood malfoy snape
turing	nondeterministic non-deterministic computability deterministic finite-state
florida	gainesville fla jacksonville tampa lauderdale
object-oriented	aspect-oriented smalltalk event-driven prolog domain-specific
dancing	singing dance dances dancers tap-dancing

Neural Word Embeddings

- We trained a language model.
- We ended up with **vector representations of words**.
- These representations are useful -- they encode various aspects of word similarity.

tagging + pre-training

we can use the **E** we got from LM training to initialize **E** for the POS tagging task.

(why is that helpful?)

NOUN

The brown **fox** jumped over

VERB

brown fox **jumped** over the

PREP

fox jumped **over** the lazy

pre-training

- This is a sort of **semi-supervised** learning or **multi-task** learning.

- We learn from "unannotated" data.

- We then use the representations on tasks with annotated data.

Neural Word Embeddings

- We trained a language model.
- We ended up with **vector representations of words**.
- These representations are useful -- they encode various aspects of word similarity.
- **A form of semi-supervised learning.**
- More next week.