Advanced Text Analysis

Day 4

Fabienne Lind & Petro Tolochko

Neural Networks & Transformer Models

Vanilla NN

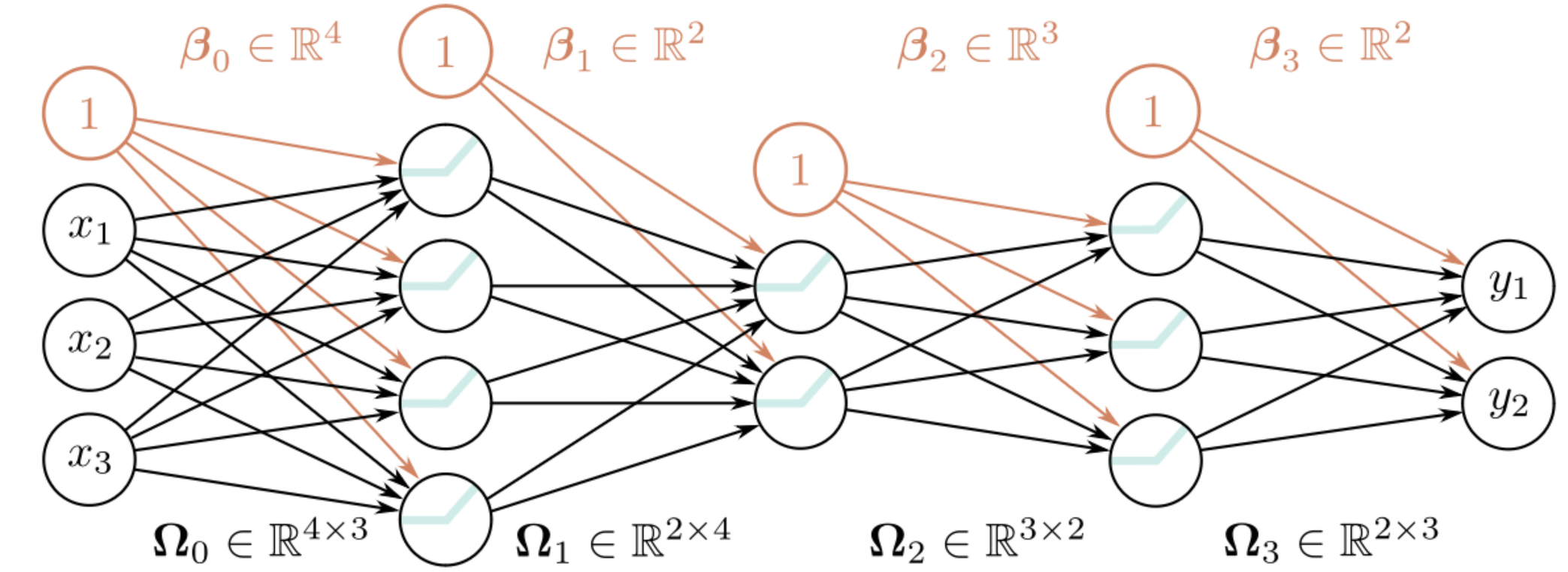
- Neural Network

Vanilla NN

- Neural Network
- Neural?

Vanilla NN

- Neural Network
- Neural?
- Network?



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

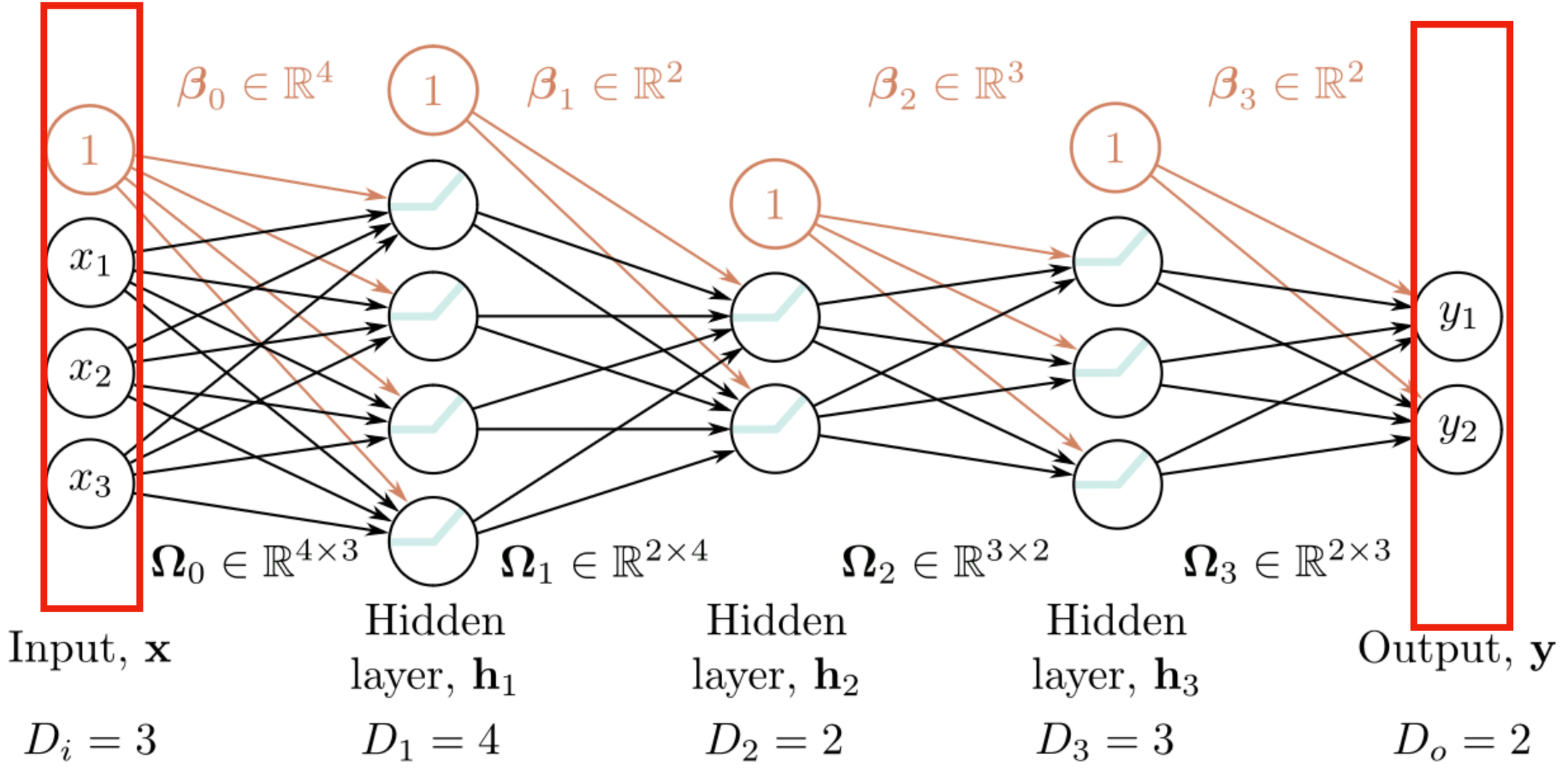
$D_2 = 2$

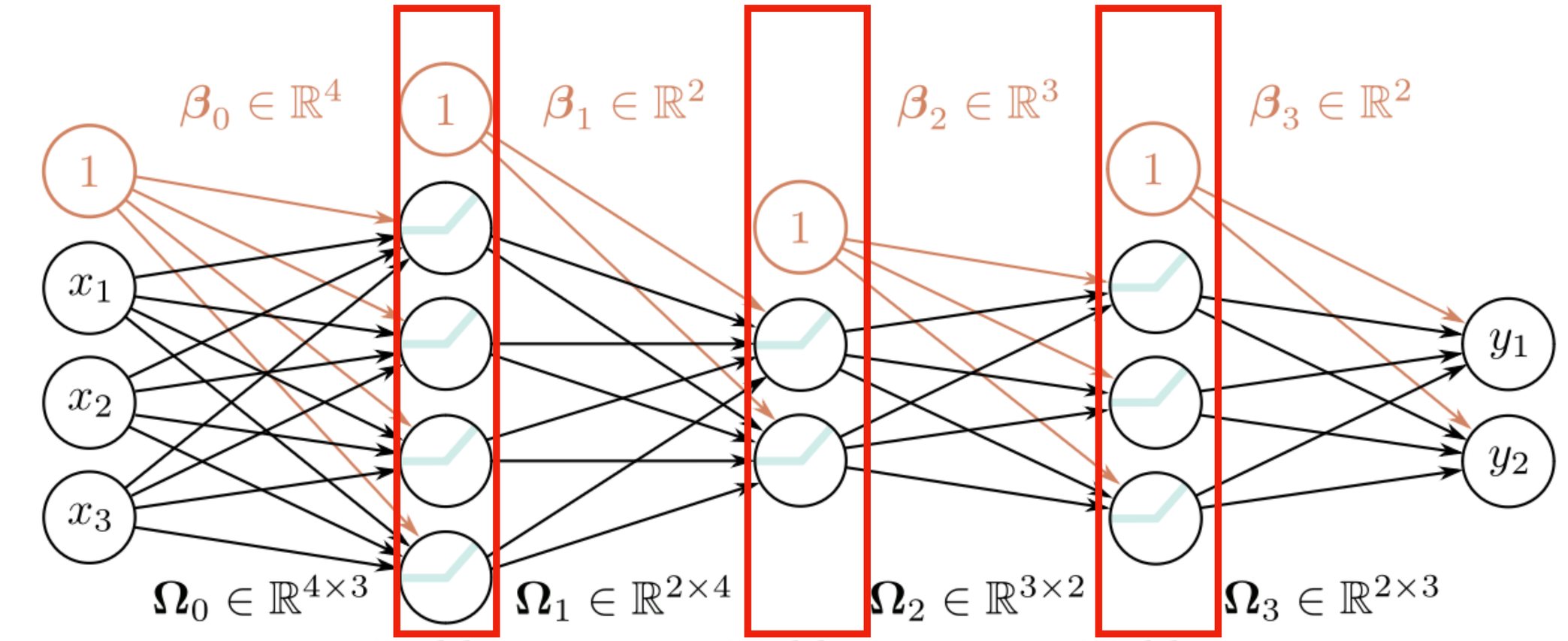
Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

Output, $\mathbf{y}$

$D_o = 2$


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

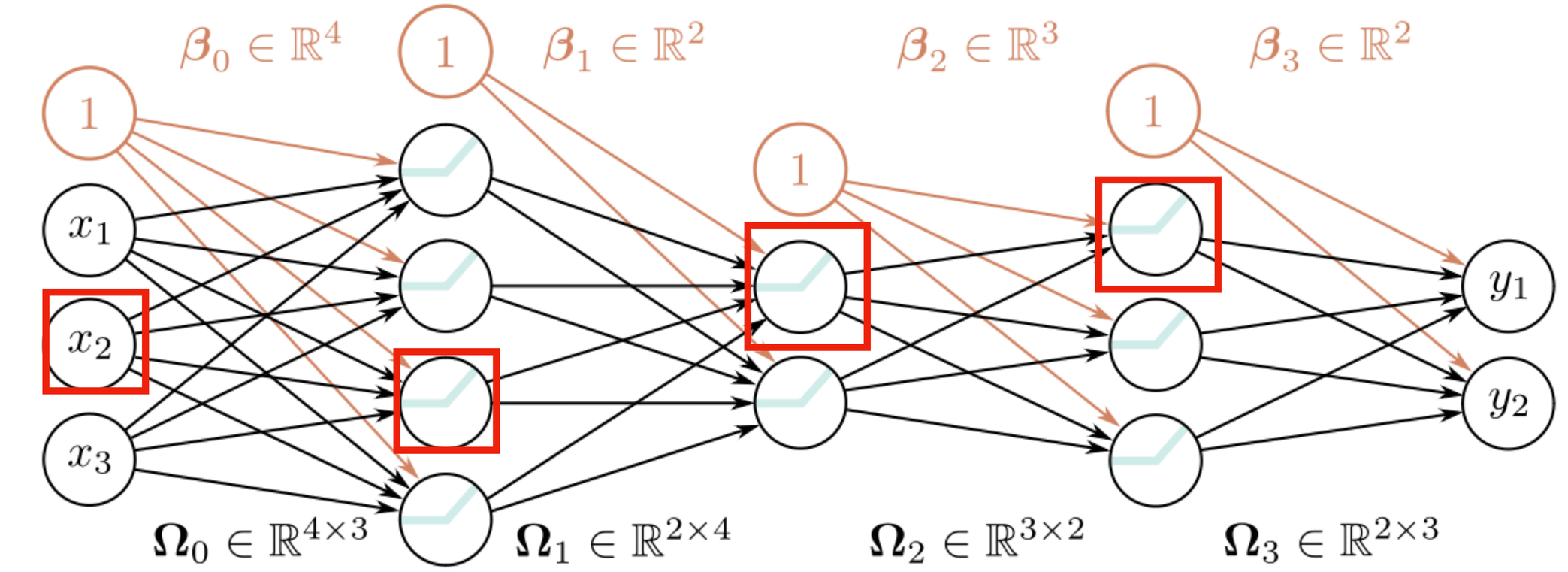
$D_2 = 2$

Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

Output, $\mathbf{y}$

$D_o = 2$



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

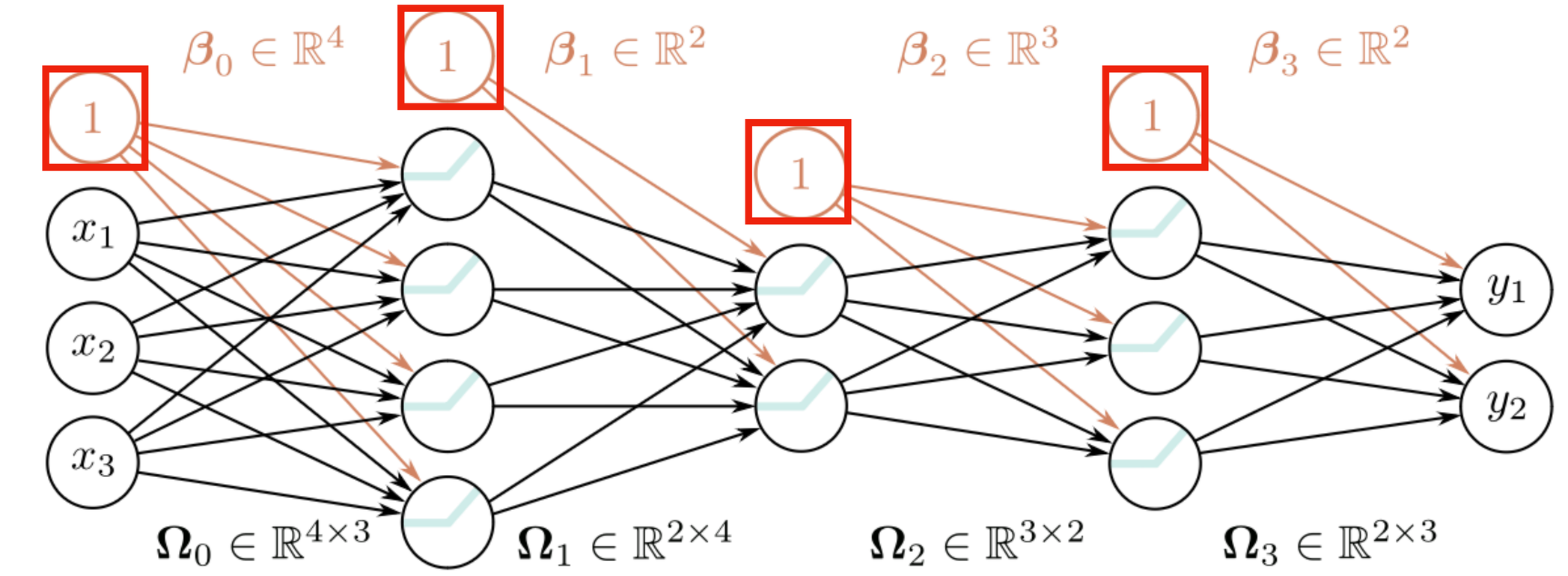
$D_2 = 2$

Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

Output, $\mathbf{y}$

$D_o = 2$



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

$D_2 = 2$

Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

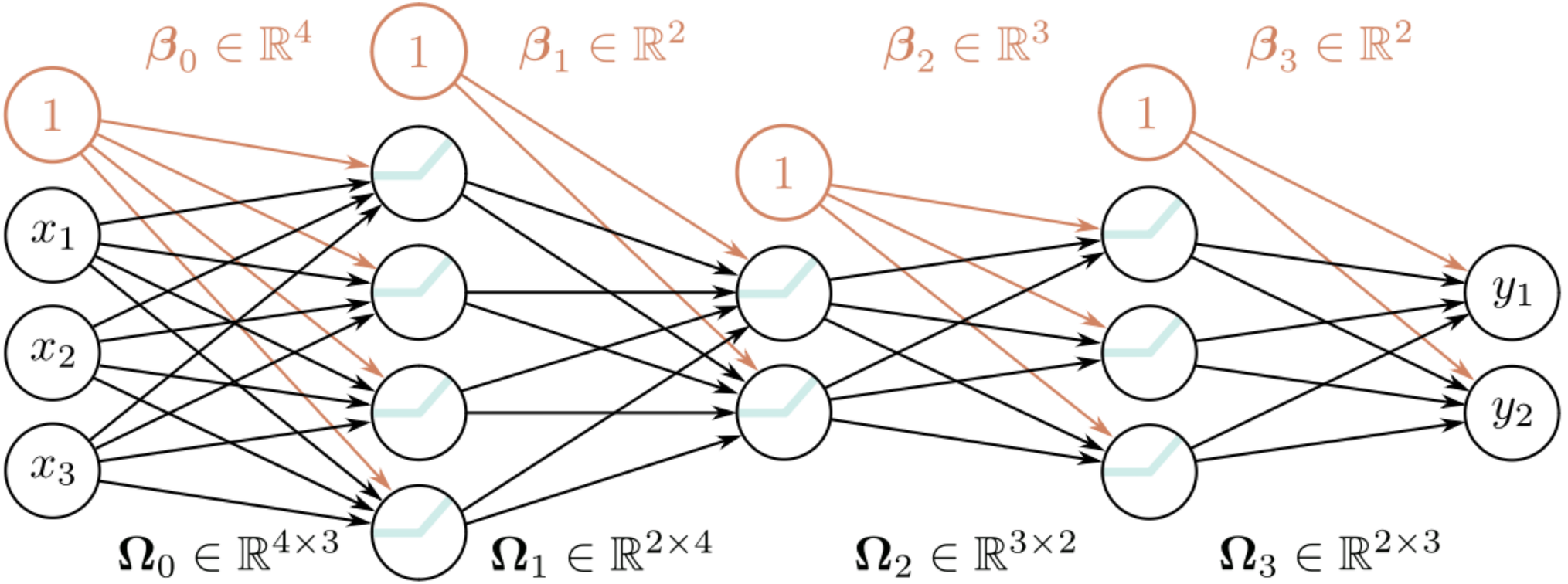
Output, $\mathbf{y}$

$D_o = 2$

Man

By

Upon



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

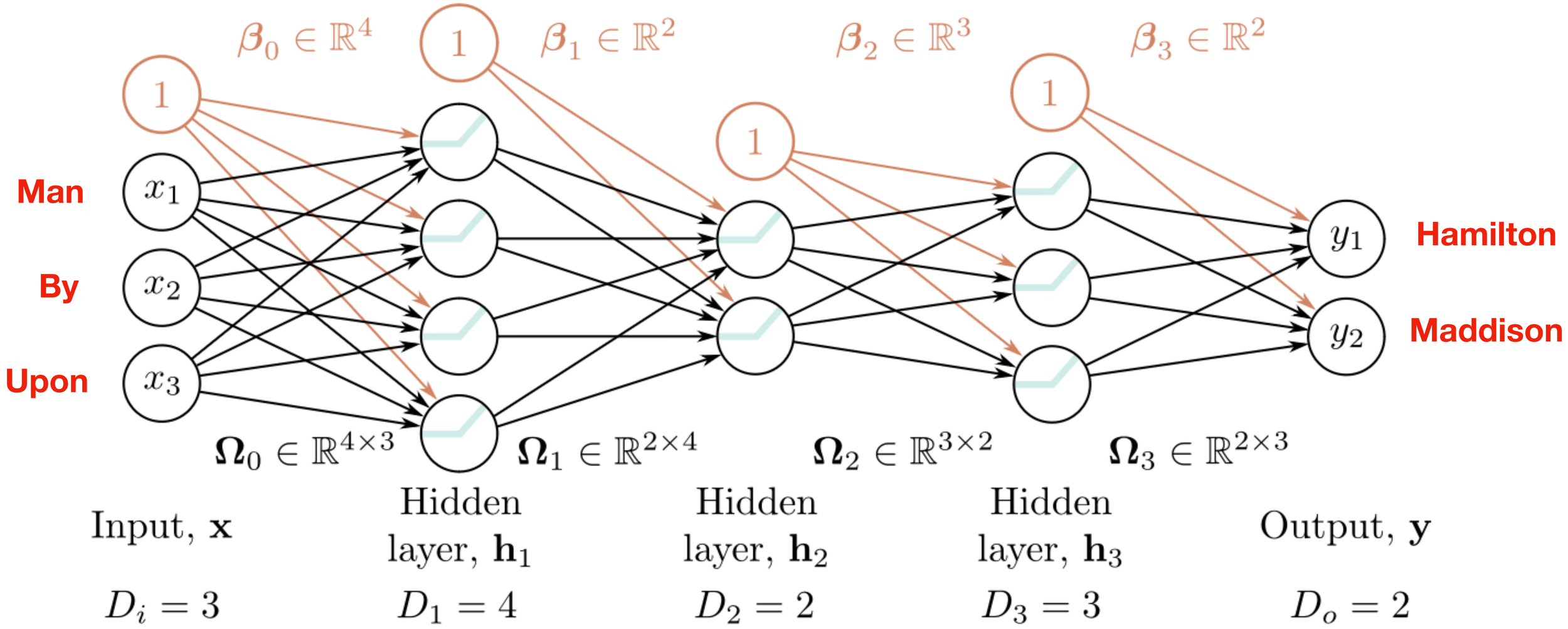
$D_2 = 2$

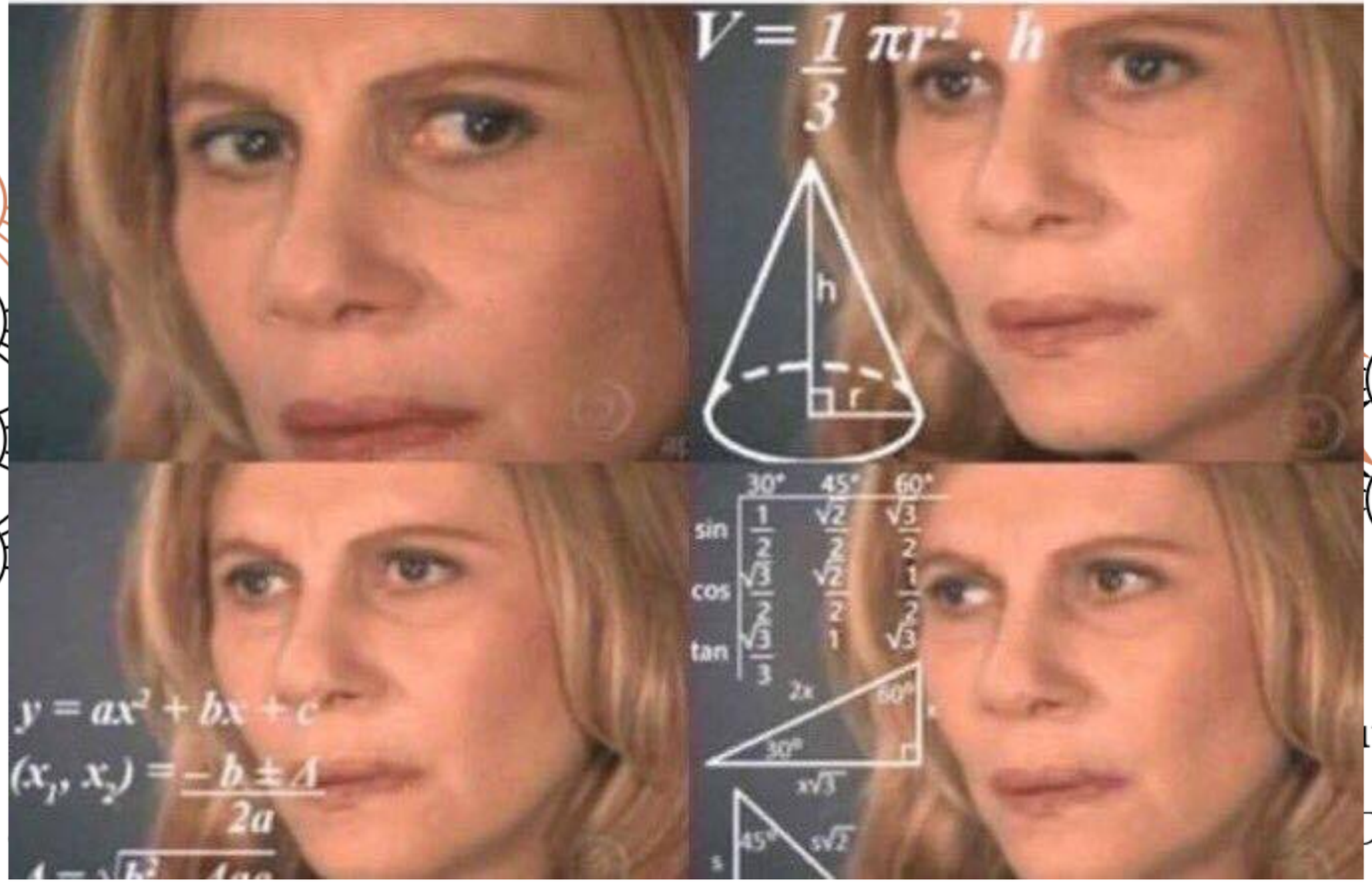
Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

Output, $\mathbf{y}$

$D_o = 2$


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
Man
By
Upon
Hamilton
Maddison
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$



Man

By

Upon

Hamilton

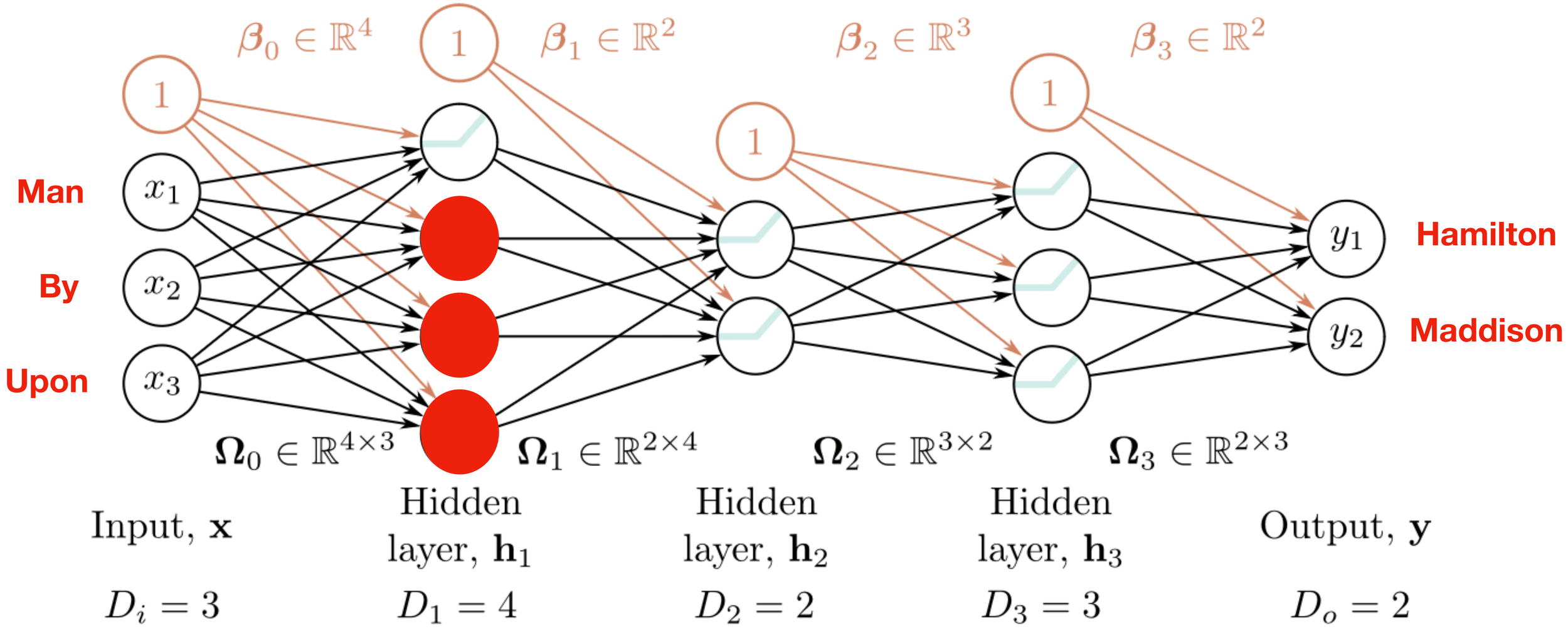
Maddison

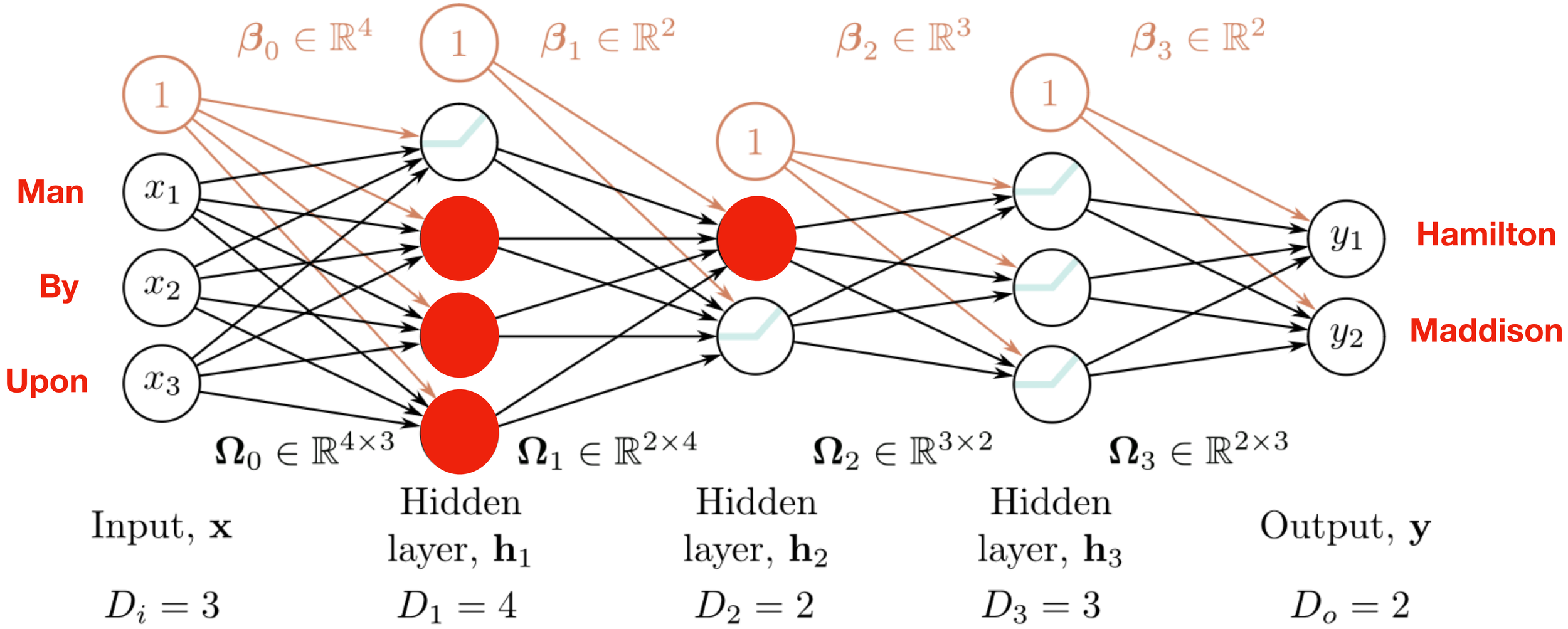
Input, $\mathbf{x}$

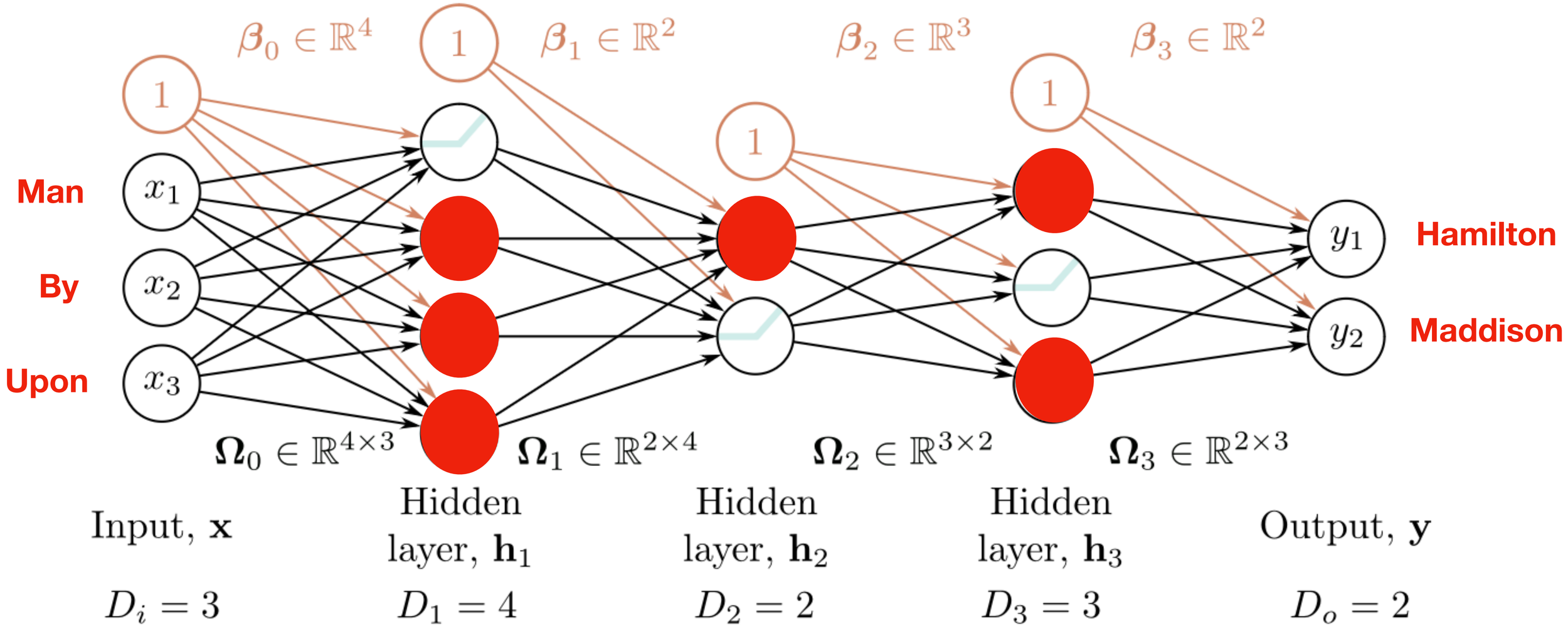
$D_i =$

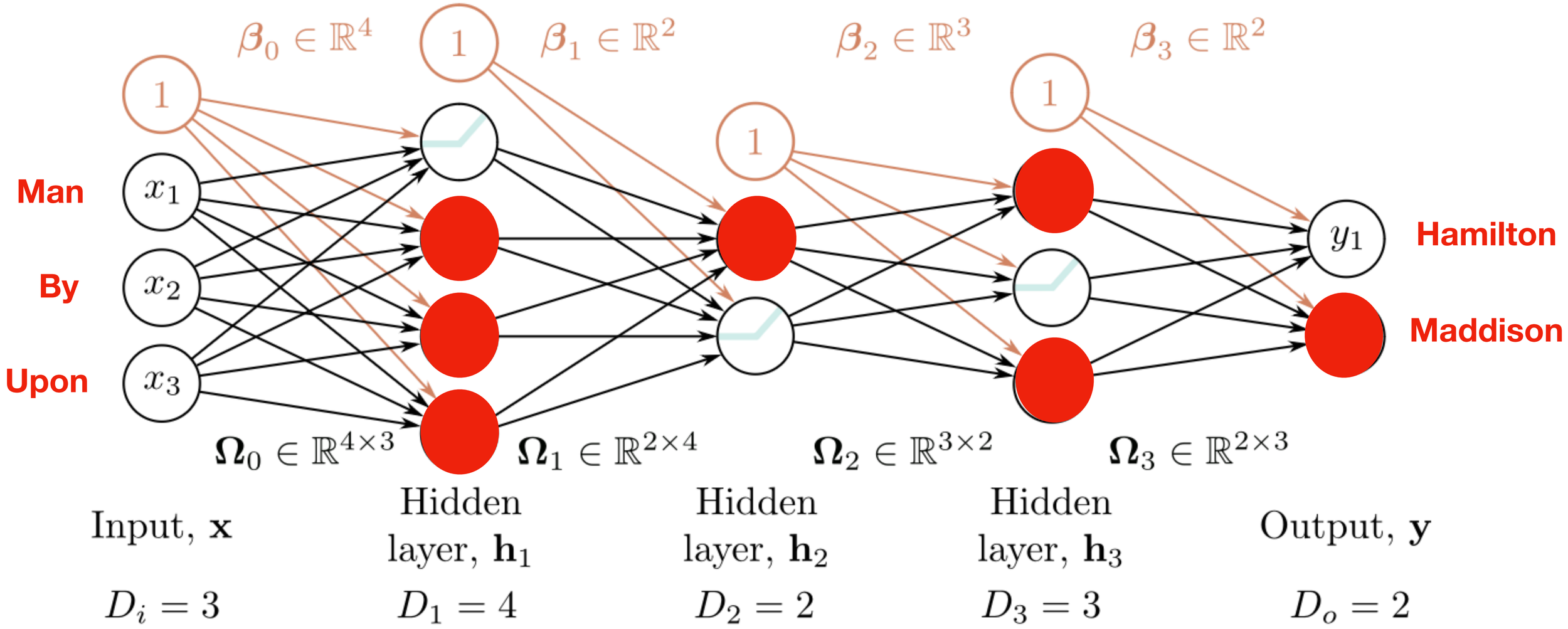
tput, $\mathbf{y}$

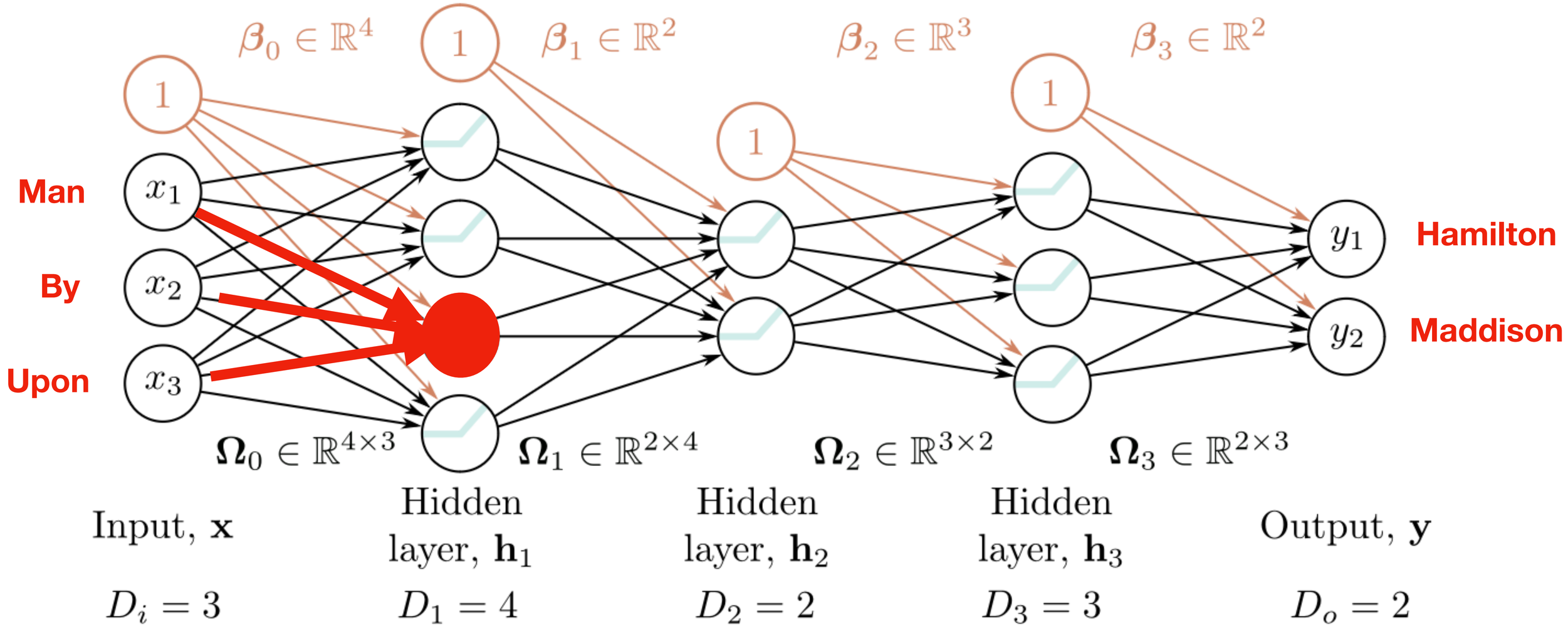
$D_o = 2$


$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
Man
By
Upon
Hamilton
Maddison
$\Omega_0 \in \mathbb{R}^{4\times3}$
$\Omega_1 \in \mathbb{R}^{2\times4}$
$\Omega_2 \in \mathbb{R}^{3\times2}$
$\Omega_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
Man
By
Upon
Hamilton
Maddison
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$


$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
Man
By
Upon
Hamilton
Maddison
$\Omega_0 \in \mathbb{R}^{4\times3}$
$\Omega_1 \in \mathbb{R}^{2\times4}$
$\Omega_2 \in \mathbb{R}^{3\times2}$
$\Omega_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
Man
By
Upon
Hamilton
Maddison
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$


$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
Man
By
Upon
Hamilton
Maddison
$\Omega_0 \in \mathbb{R}^{4\times3}$
$\Omega_1 \in \mathbb{R}^{2\times4}$
$\Omega_2 \in \mathbb{R}^{3\times2}$
$\Omega_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + \ldots + w_n a_n$$

$$\boxed{w_1 a_1 + w_2 a_2 + w_3 a_3} + \ldots + w_n a_n$$

Activation of a Neuron

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + \ldots + w_n a_n$$

Activation of a Neuron

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + \ldots + w_n a_n \longrightarrow [-\infty; \infty]$$

Activation of a Neuron

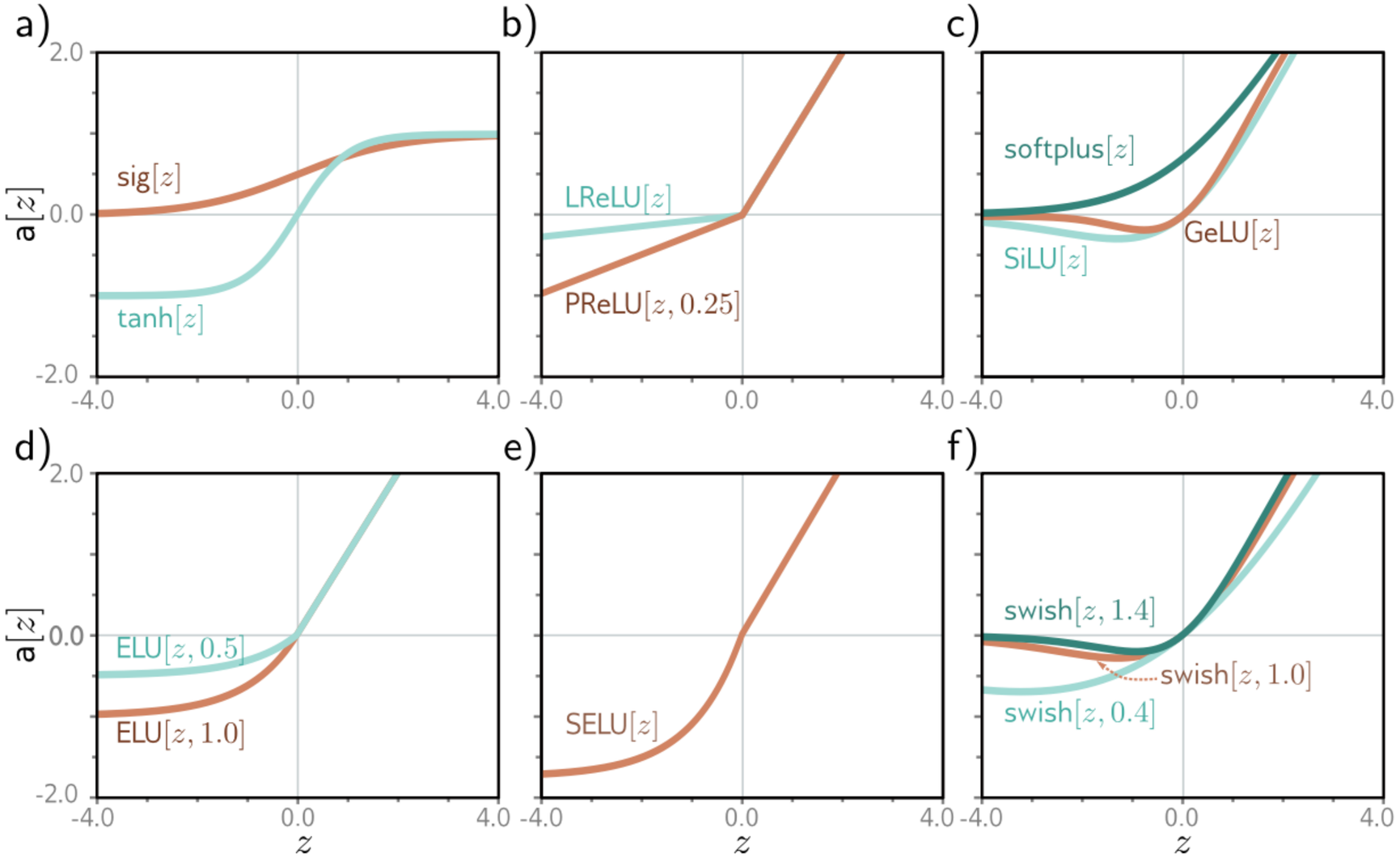
$$w_1 a_1 + w_2 a_2 + w_3 a_3 + \ldots + w_n a_n \longrightarrow f(\cdot) \qquad [-\infty; \infty]$$

Activation of a Neuron

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + \ldots + w_n a_n \longrightarrow f(\,\cdot\,) \longrightarrow$$ Something more Meaningful

Activation of a Neuron

Activation of a Neuron



Prince, 2023

Figure 3.13 Activation functions. a) Logistic sigmoid and tanh functions. b) Leaky ReLU and parametric ReLU with parameter 0.25. c) SoftPlus, Gaussian error linear unit, and sigmoid linear unit. d) Exponential linear unit with parameters 0.5 and 1.0, e) Scaled exponential linear unit. f) Swish with parameters 0.4, 1.0, and 1.4.

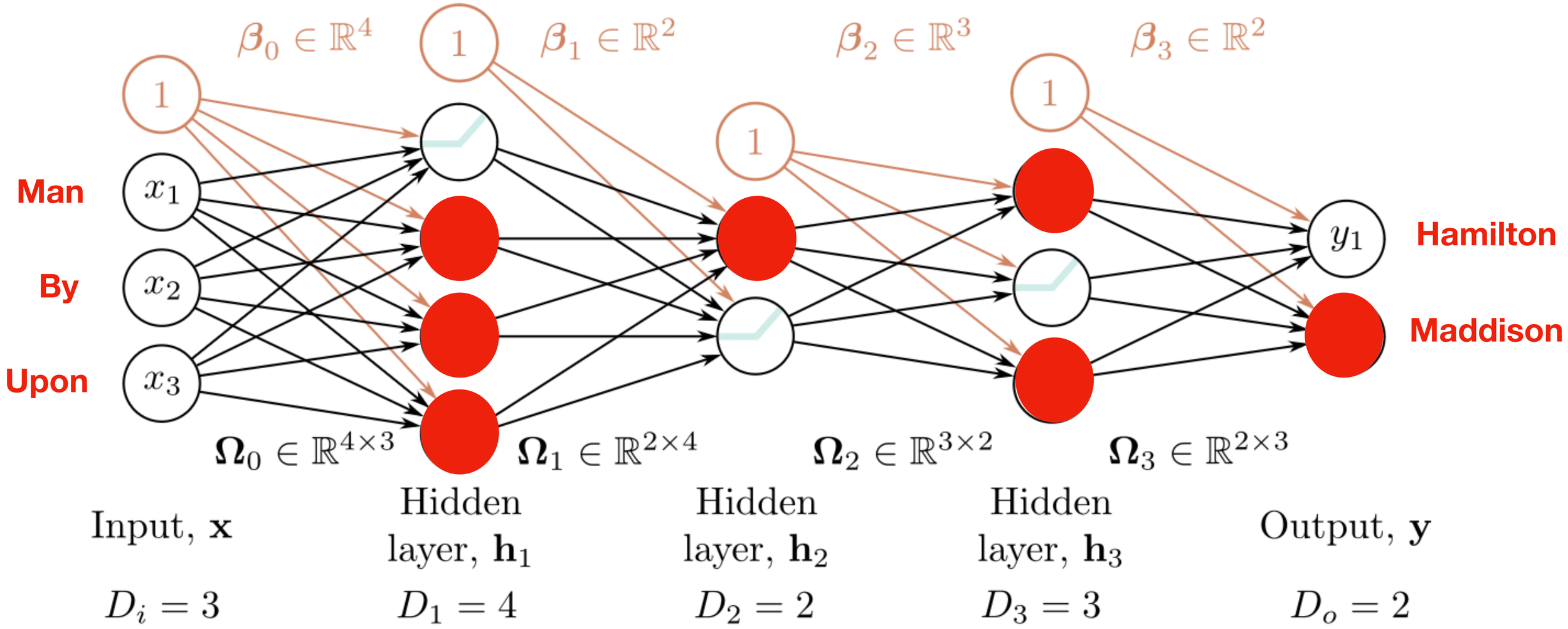
Activation of a Neuron

$$\sigma(w_1 a_1 + w_2 a_2 + \ldots + w_n a_n + \beta)$$

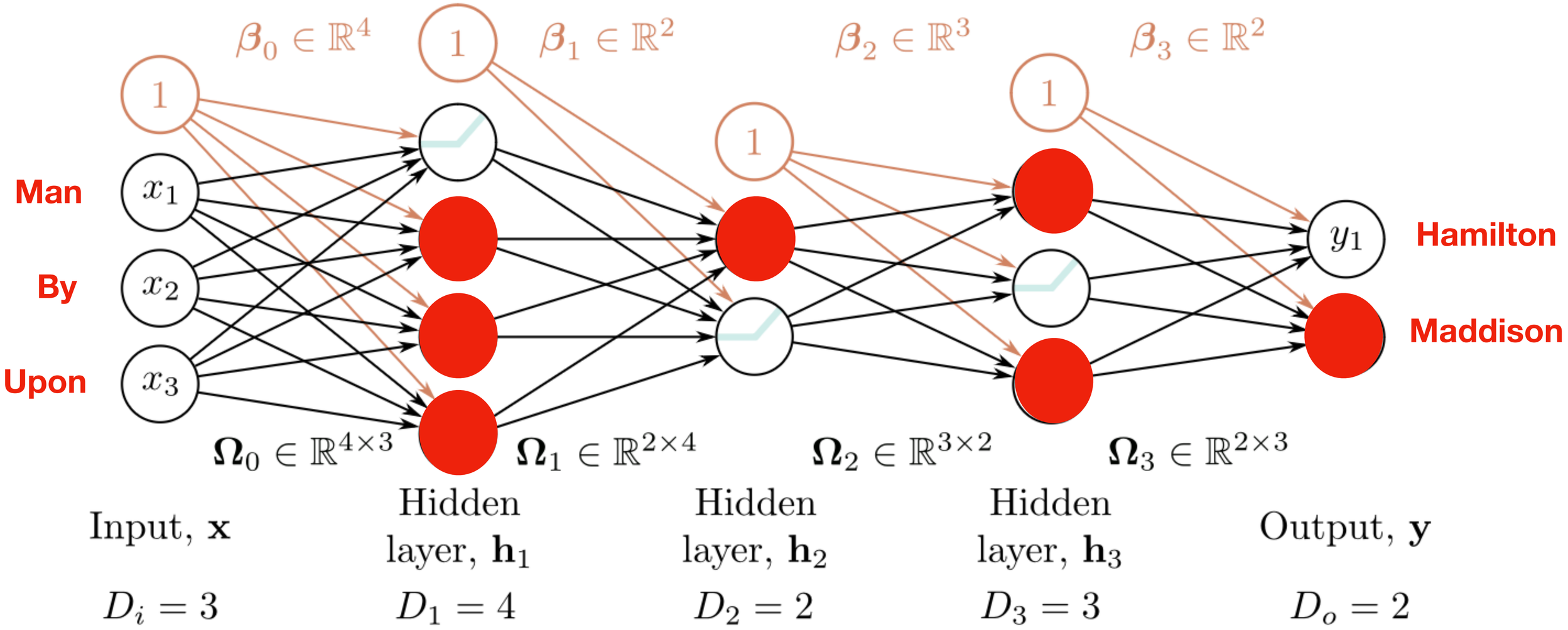
Activation of a Neuron

$$\sigma(w_1 a_1 + w_2 a_2 + \ldots + w_n a_n + \beta)$$

$$f(W^T A + \beta)$$

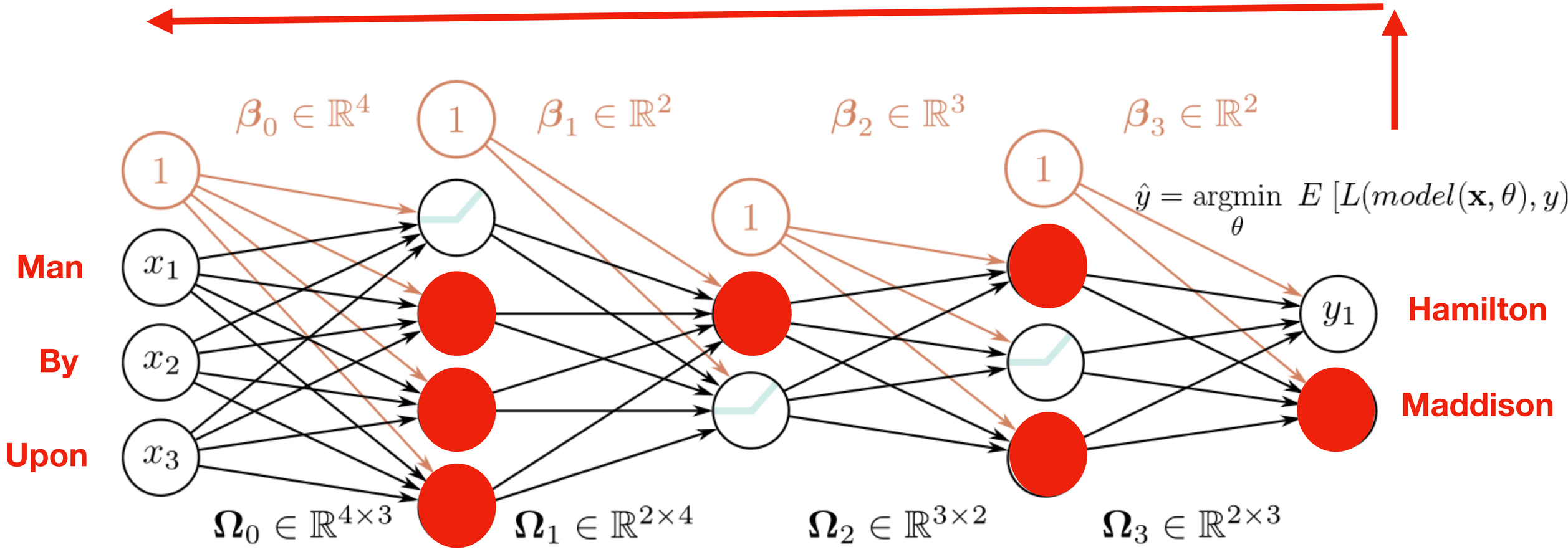

$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
Man
By
Upon
Hamilton
Maddison
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

Random Assignment of Weights



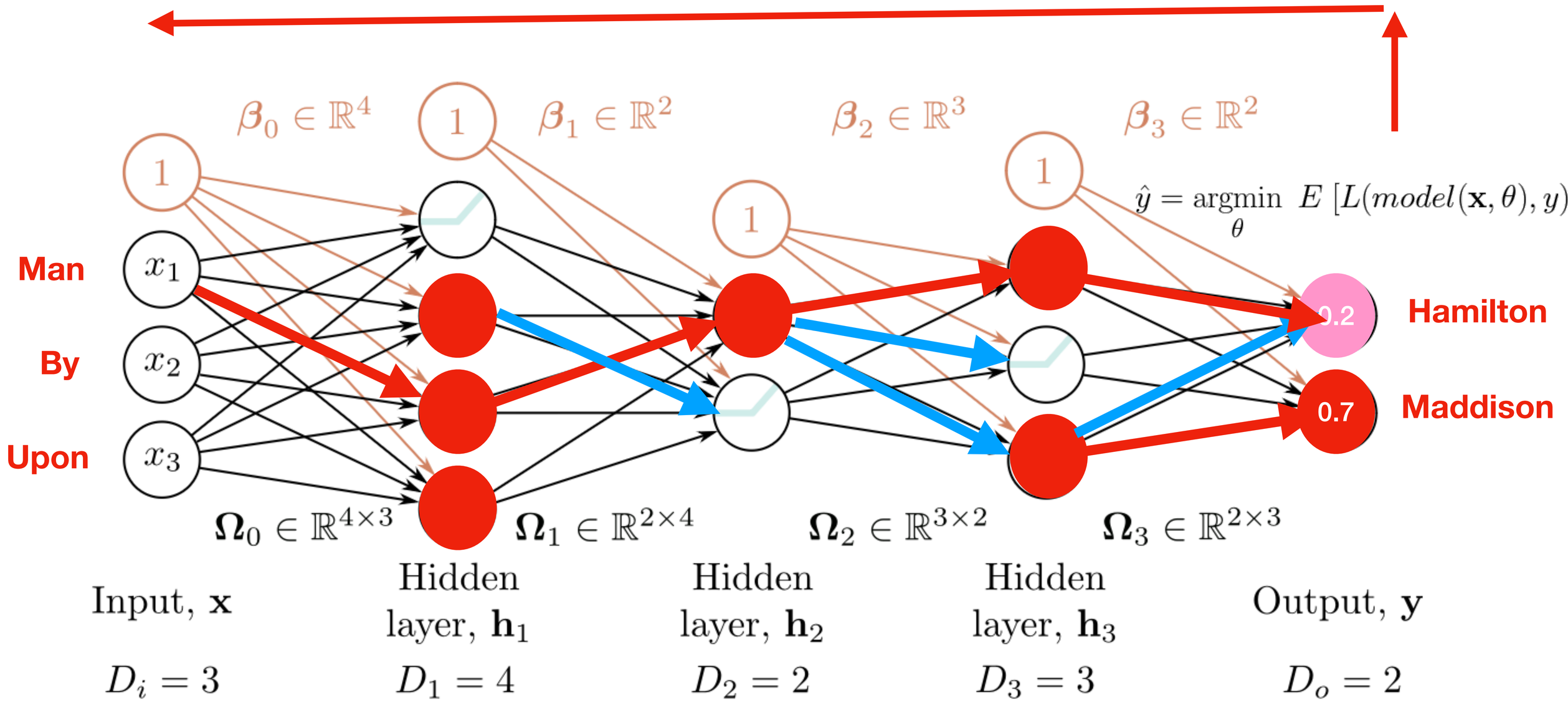
Loss Functions

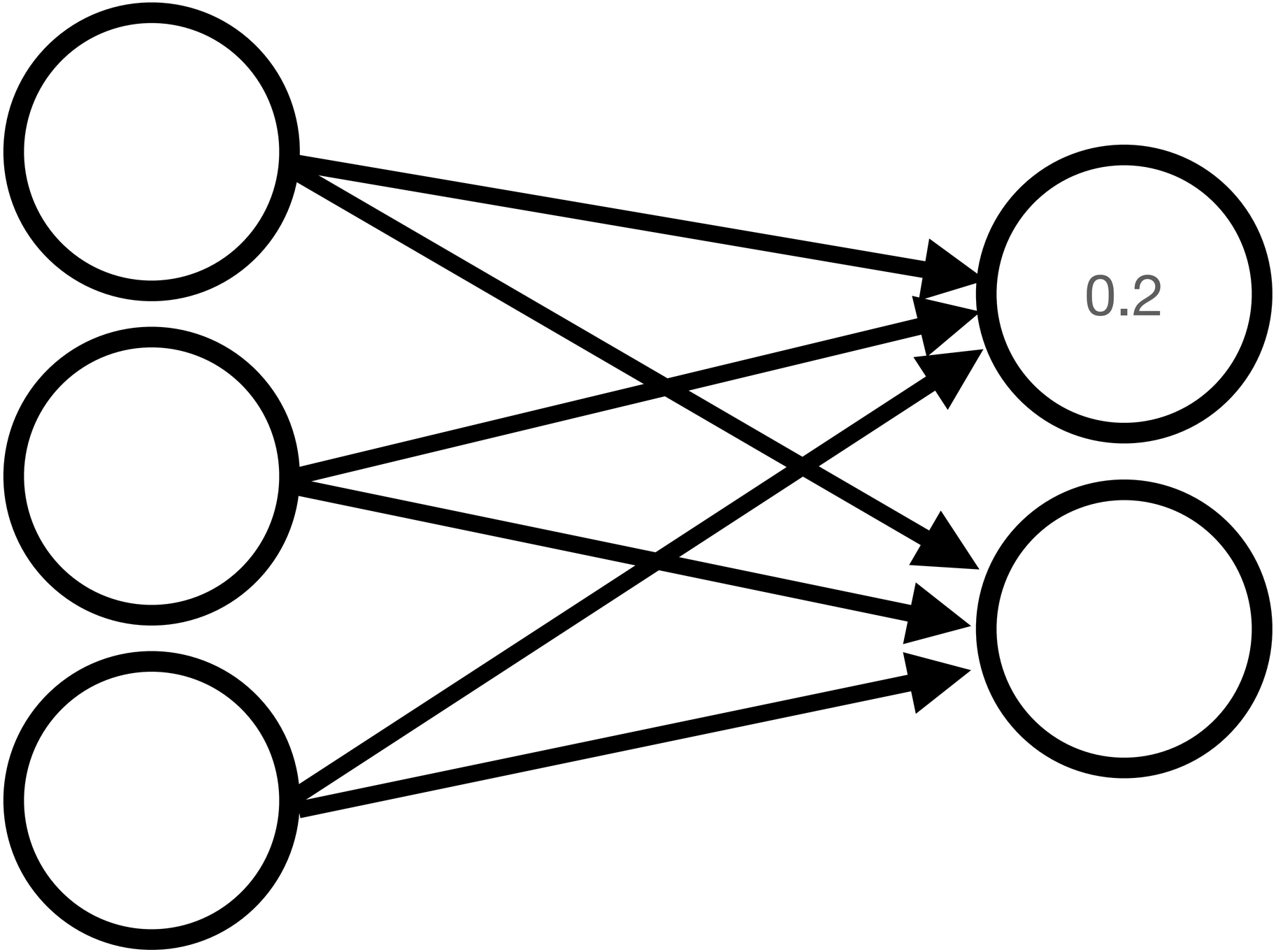
$$\hat{y} = \underset{\theta}{\operatorname{argmin}} \; E\left[L(model(\mathbf{x}, \theta), y)\right]$$

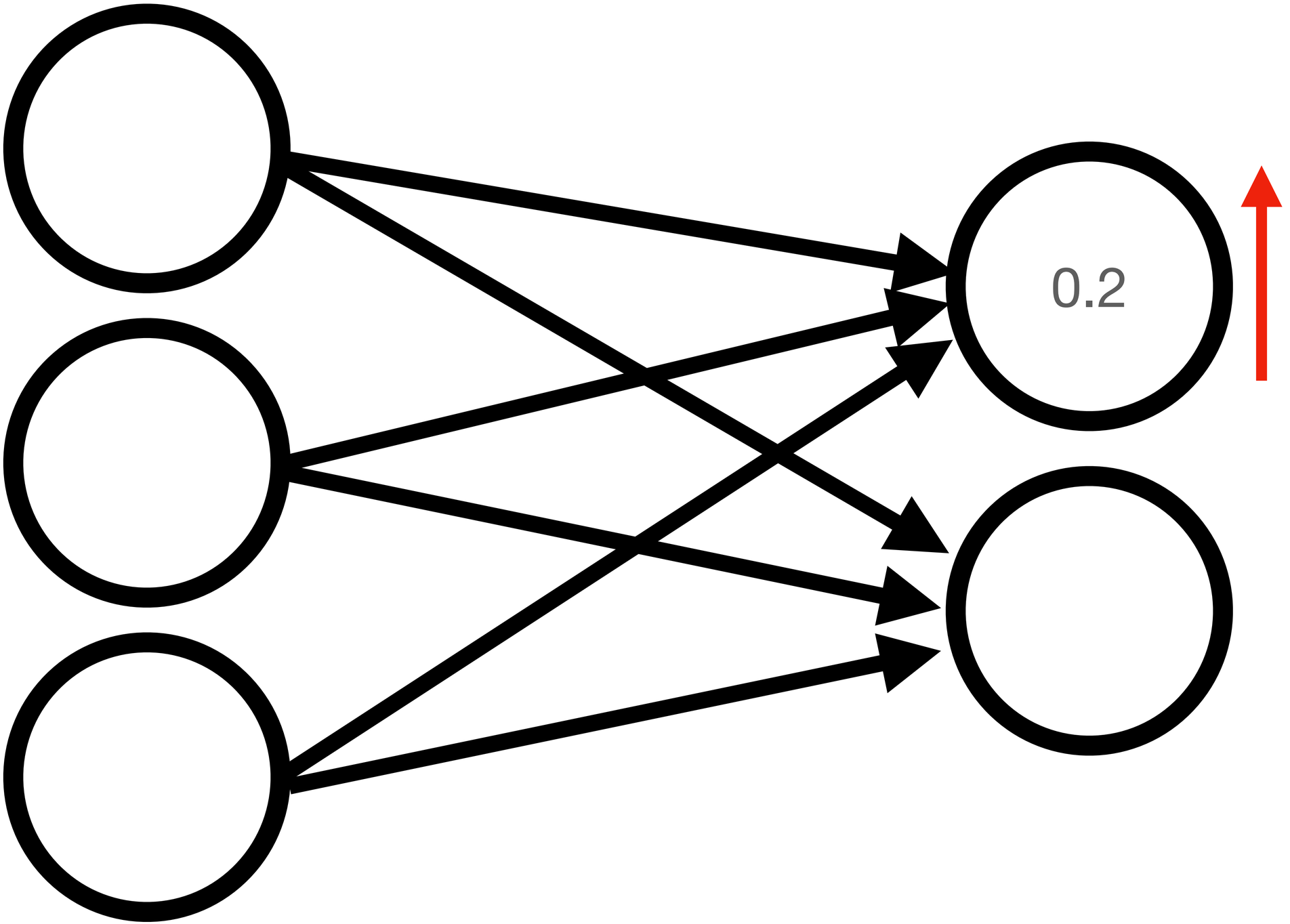

$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
$\hat{y} = \underset{\theta}{\text{argmin}}\ E\left[L(model(\mathbf{x}, \theta), y)\right]$
y_1
Man
By
Upon
Hamilton
Maddison
$\mathbf{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\mathbf{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\mathbf{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\mathbf{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

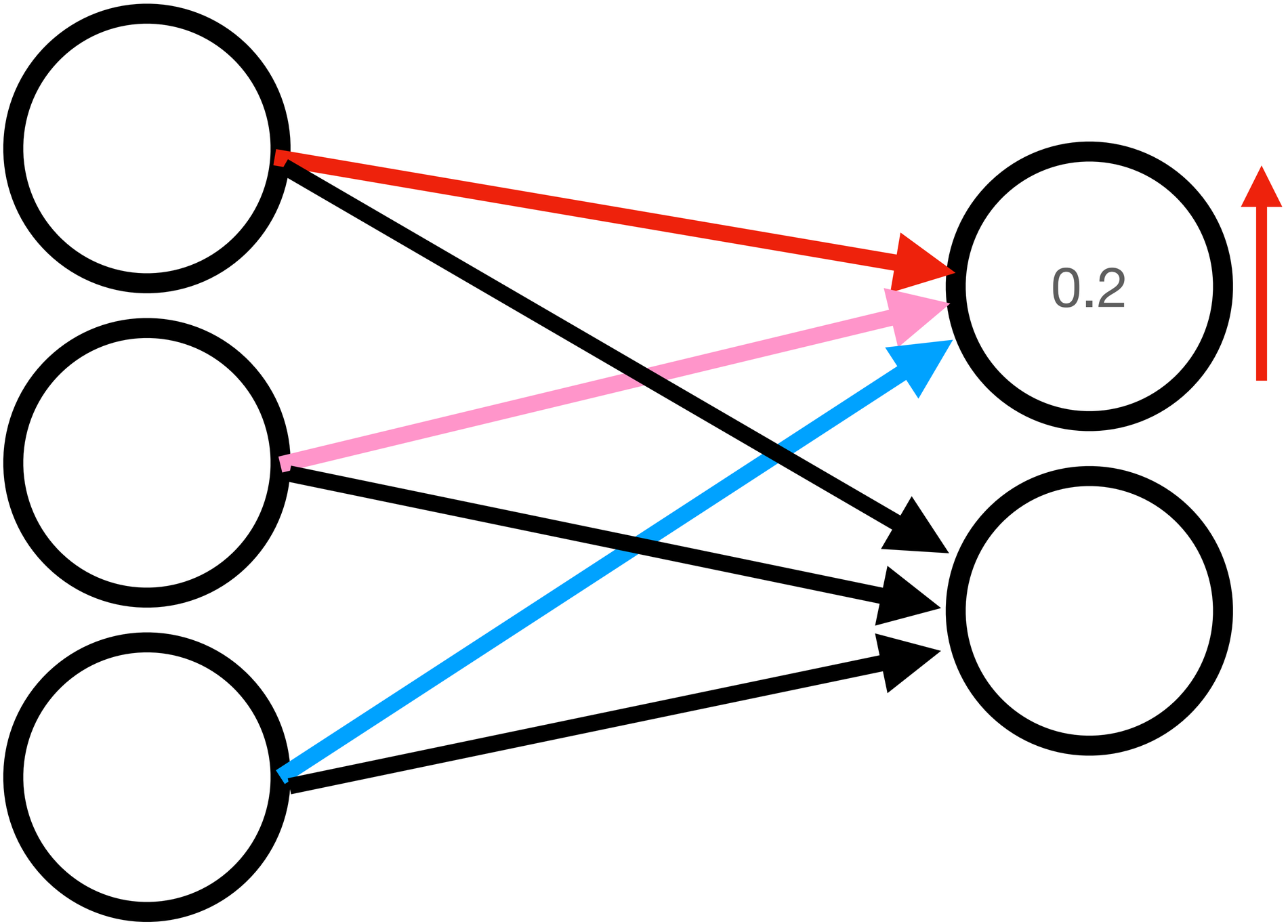
Backpropagation

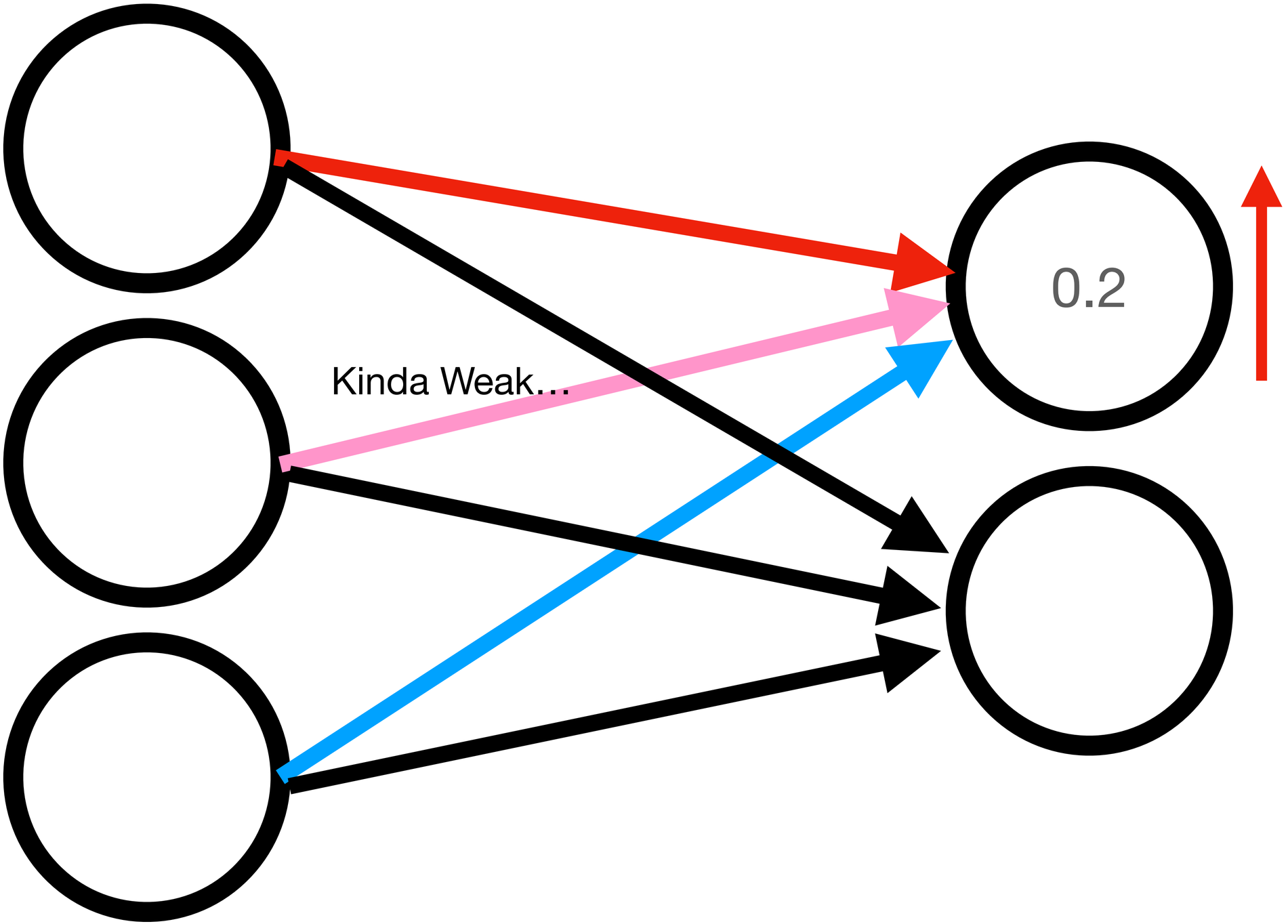
- We need to propagate the gradient back through the network
- Evaluate the gradient of a loss function with the respect to weights
- Recursive application of Chain Rule

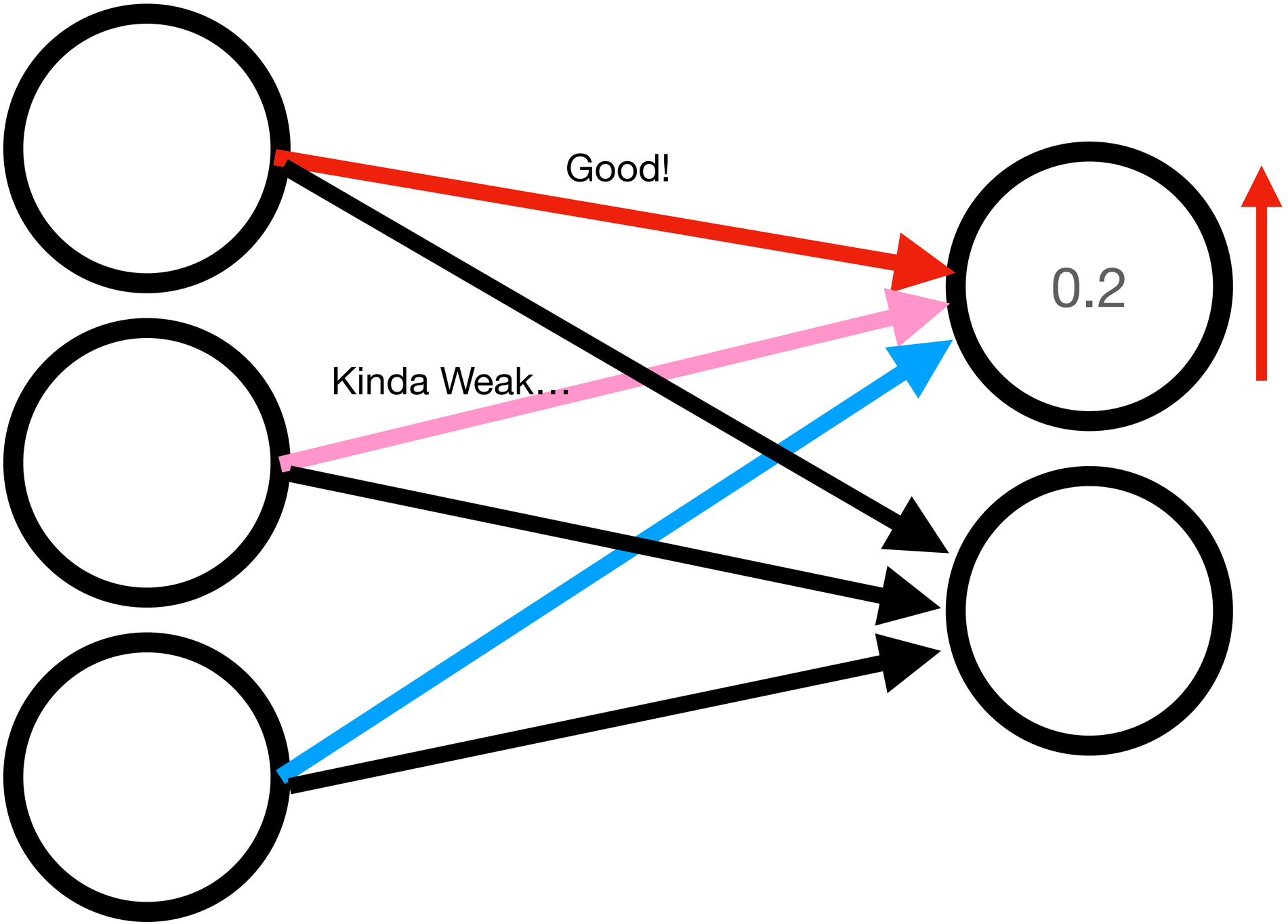

$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
1
1
1
$\hat{y} = \underset{\theta}{\text{argmin}}\ E\left[L(model(\mathbf{x}, \theta), y)\right]$
Man
By
Upon
x_1
x_2
x_3
0.2
0.7
Hamilton
Maddison
$\mathbf{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\mathbf{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\mathbf{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\mathbf{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

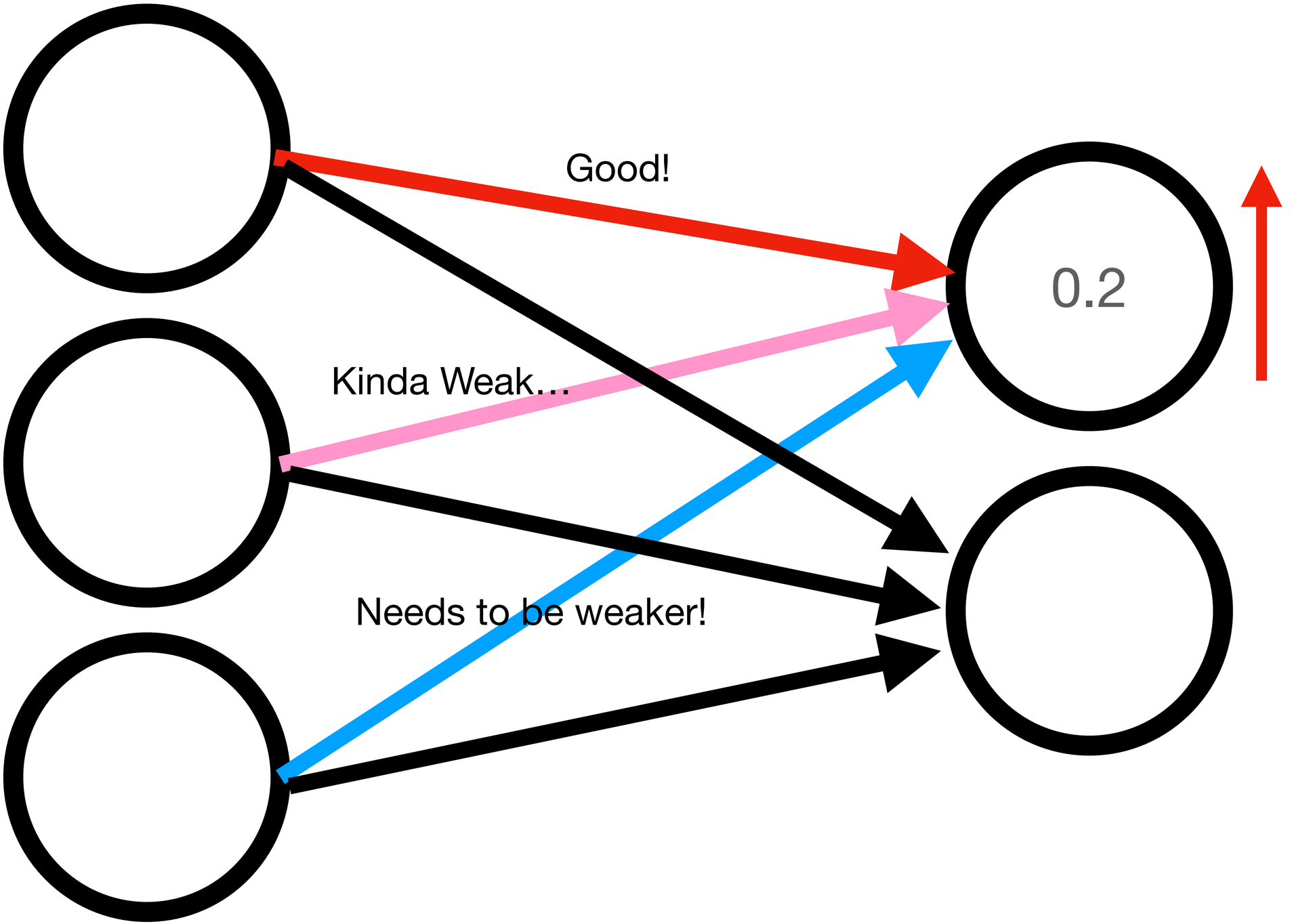

0.2

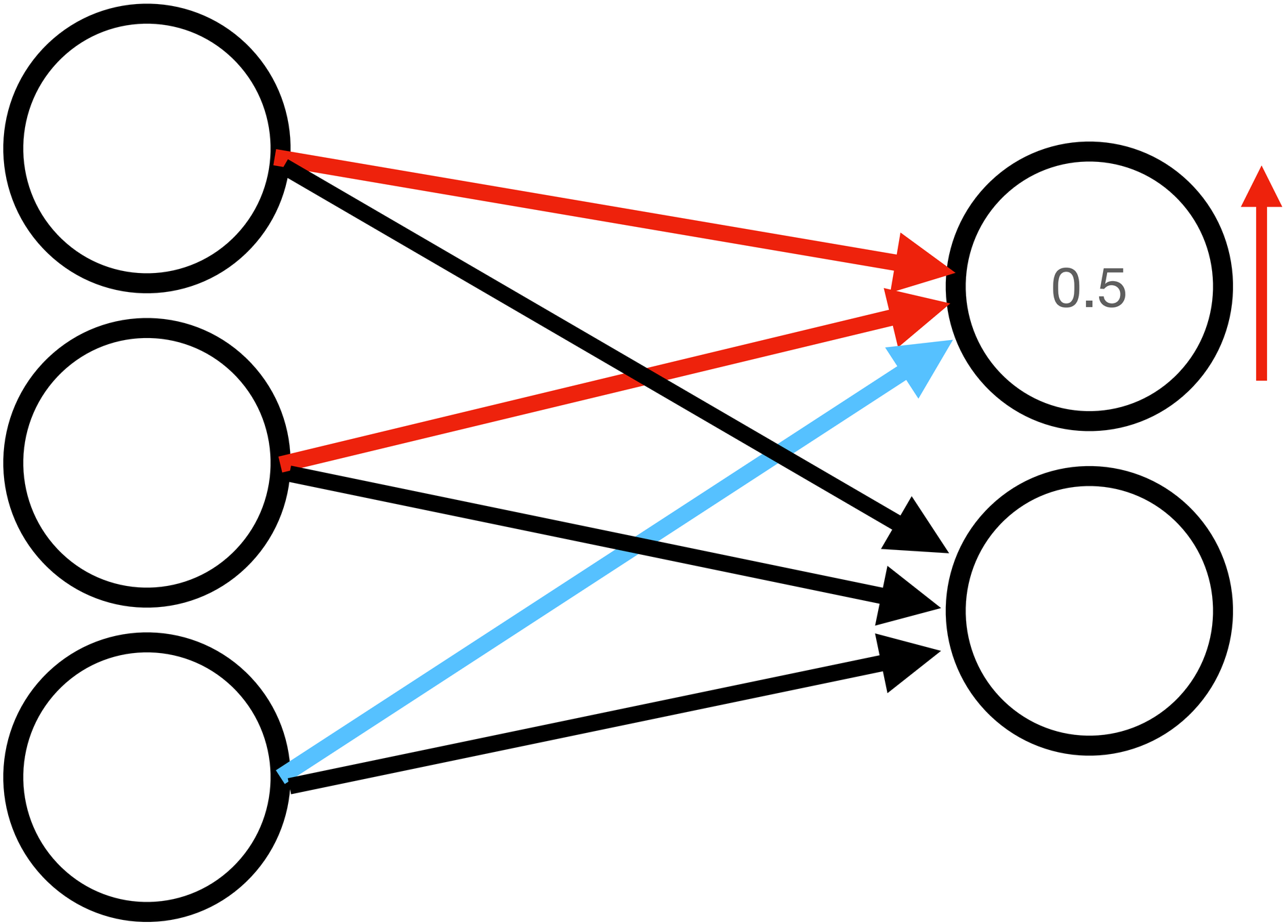

0.2

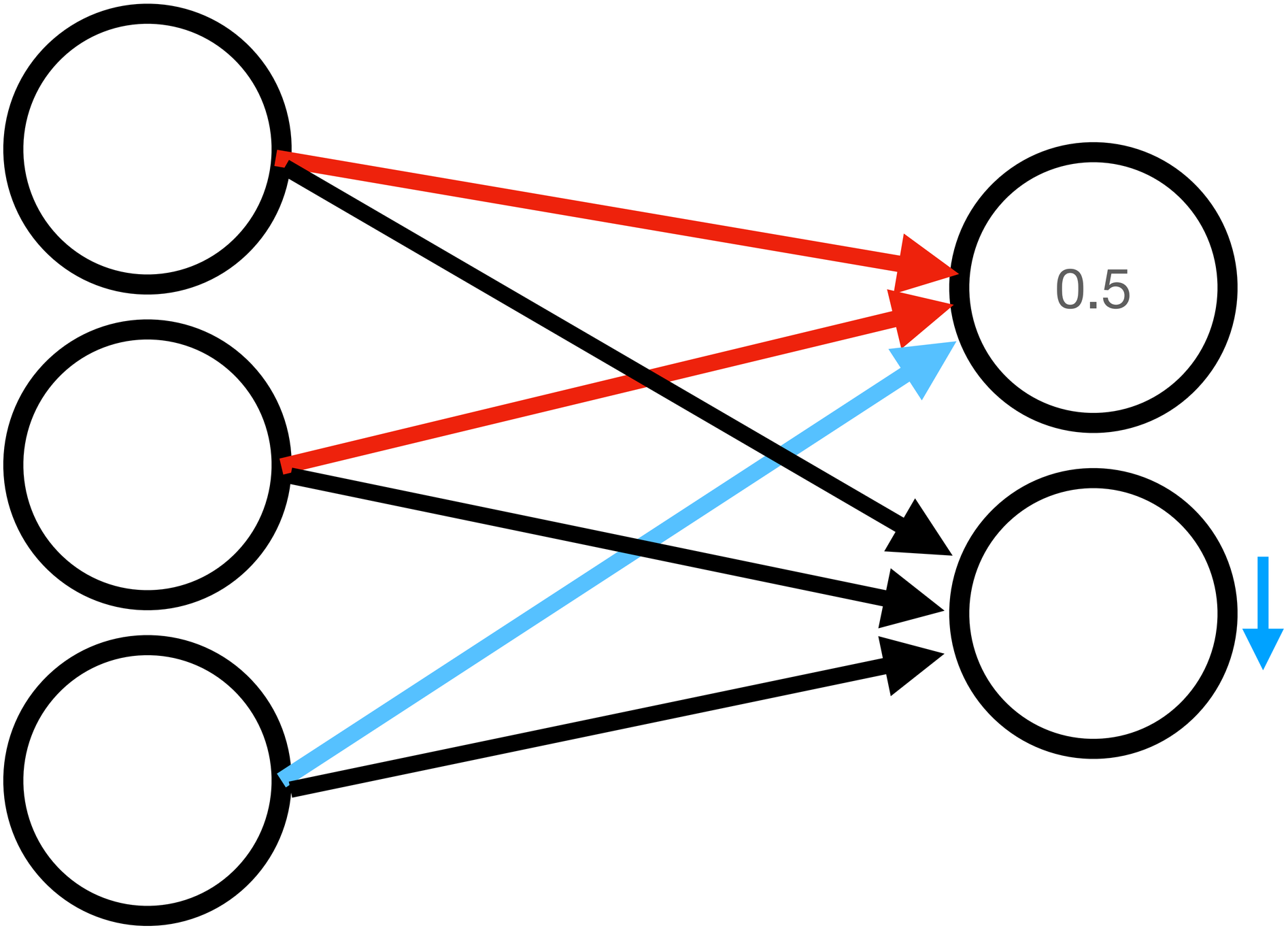

0.2

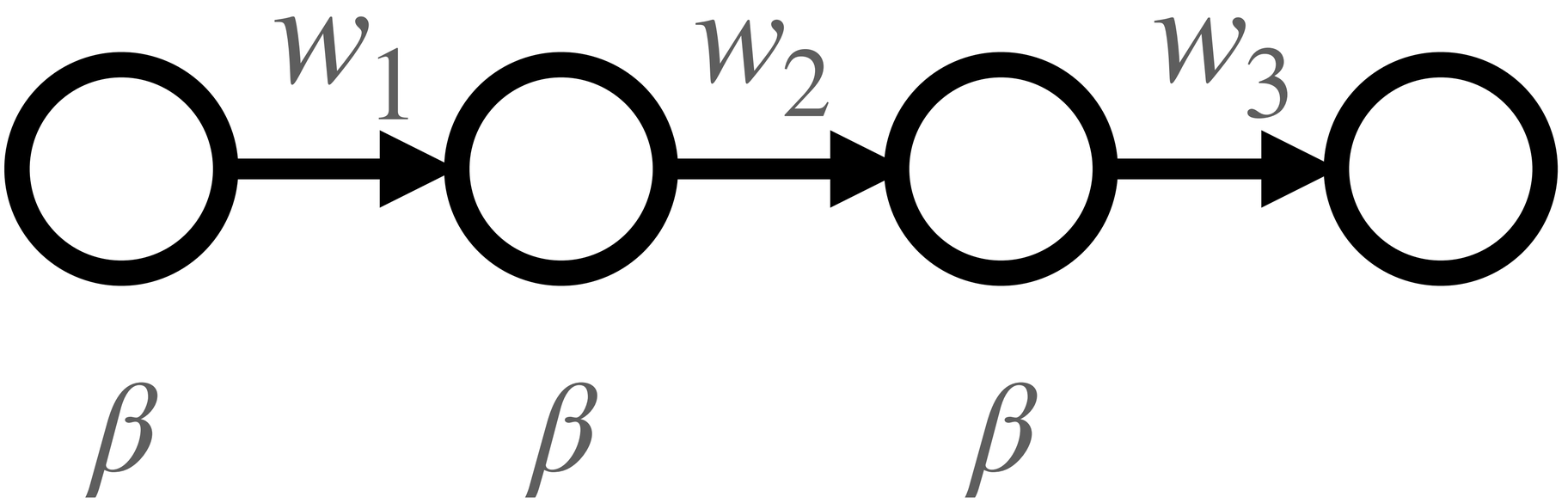

0.2
Kinda Weak…


Good!
Kinda Weak…
0.2

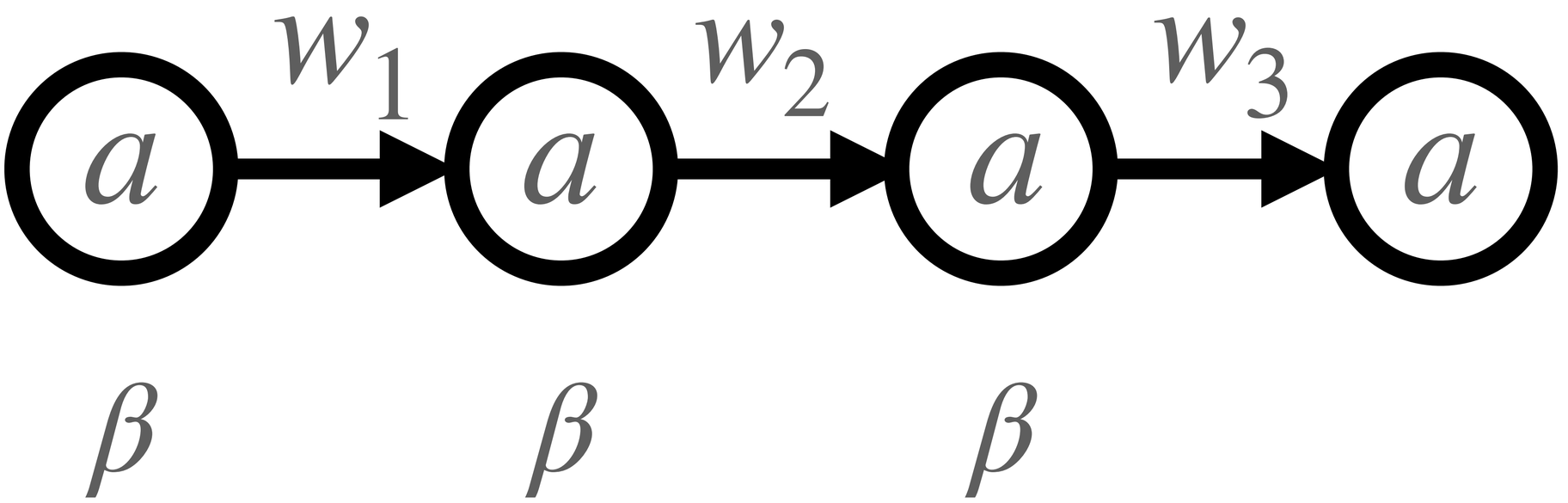

Good!
0.2
Kinda Weak…
Needs to be weaker!


0.5

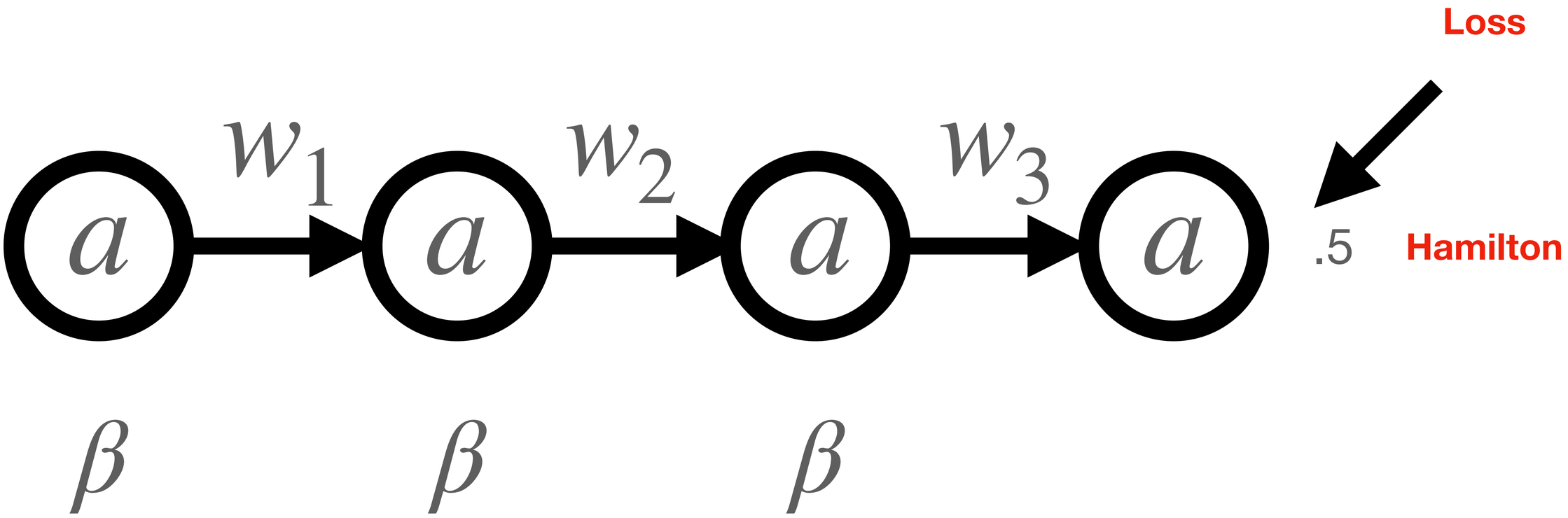

0.5

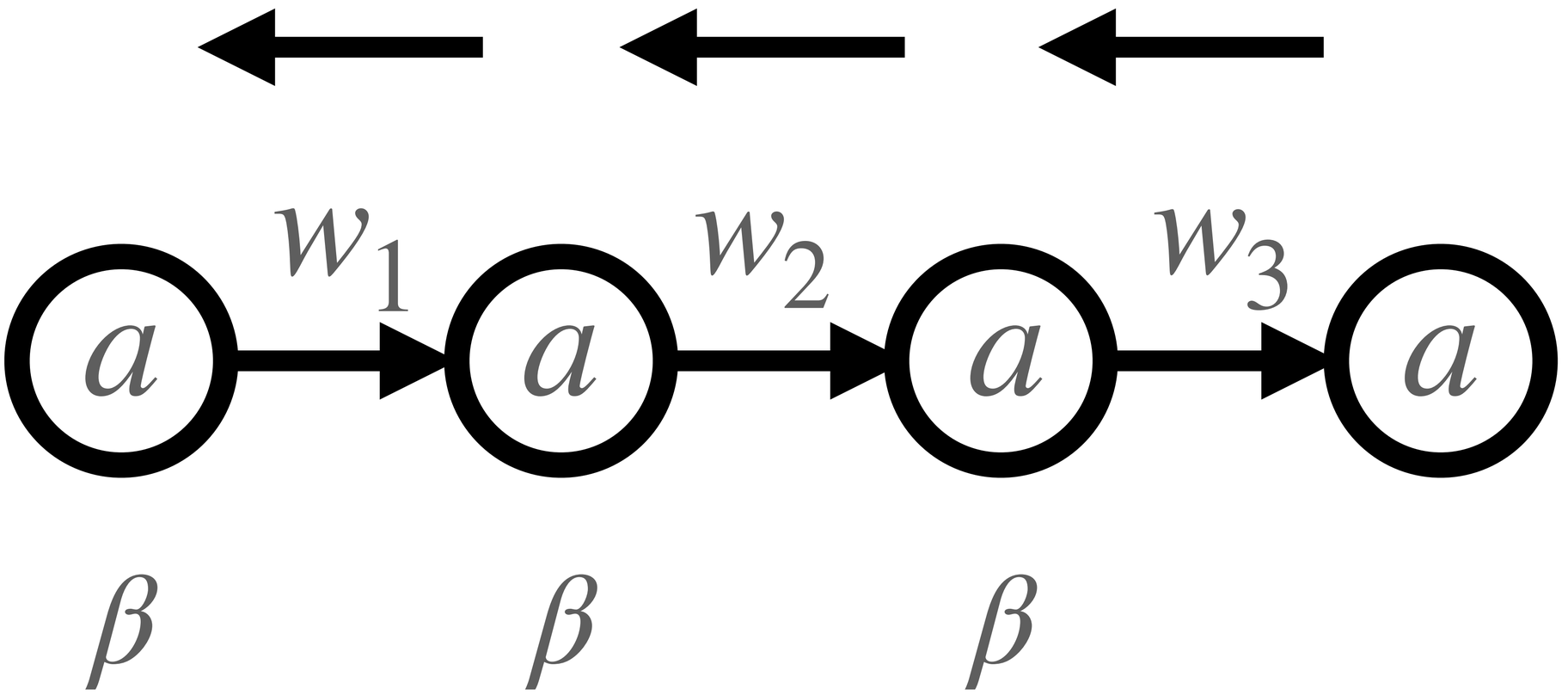


Hamilton



Hamilton


Loss
w_1
w_2
w_3
a
a
a
a
.5
Hamilton
β
β
β



.5 **Hamilton**

Questions?

Embeddings

Embeddings

- a dense numerical representation of objects in a continuous vector space

Embeddings

- a numerical representation of objects in a continuous vector space
- words, sentences, documents, etc.

Embeddings

- traditional approaches to representing objects, such as one-hot encoding, are not suitable for NLP tasks due to their high dimensionality and lack of semantic information

Embeddings

- embeddings aim to capture the essence of an object by mapping it to a lower-dimensional vector space, where proximity in the vector space reflects similarity

Continuous Vector Space

- embeddings map objects to a continuous vector space, typically of lower dimensionality compared to the original space

Continuous Vector Space

- embeddings map objects to a continuous vector space, typically of lower dimensionality compared to the original space
- each dimension in the vector space represents a learned feature or property of the object
- proximity or distance between vectors = similarity
- representing objects as vectors, various mathematical operations and computations can be performed on the embeddings for prediction and/or inference

Word Embeddings

- words are represented as dense vectors in the embedding space, where words with similar meanings or contexts are located closer to each other
- word embeddings can capture semantic relationships
- sentiment analysis
- machine translation
- document classification
- Etc.

Word2Vec

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Continuous Bag-of-Words Model

Continuous Skip-gram Model

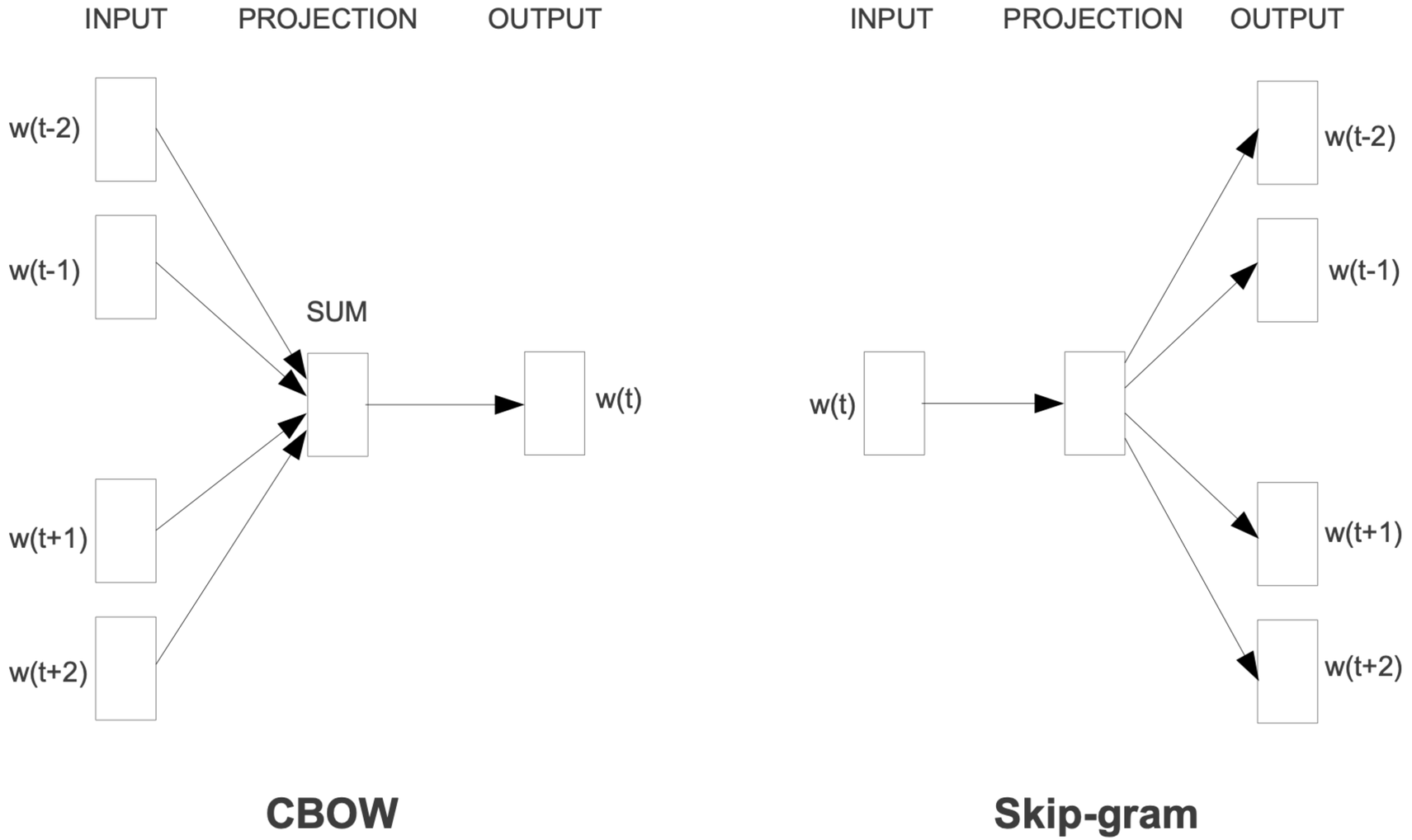
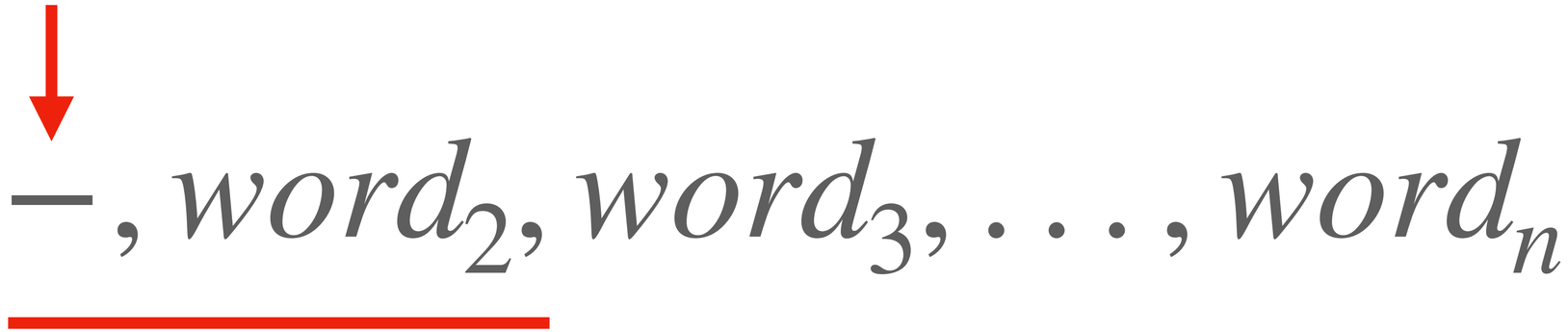


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

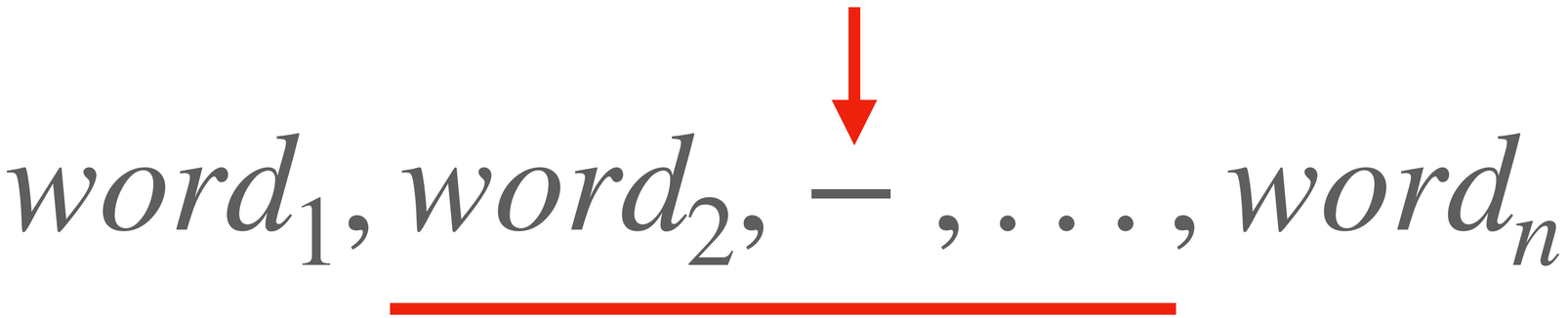
Continuous Bag-of-Words Model

$$word_1, word_2, word_3, \ldots, word_n$$

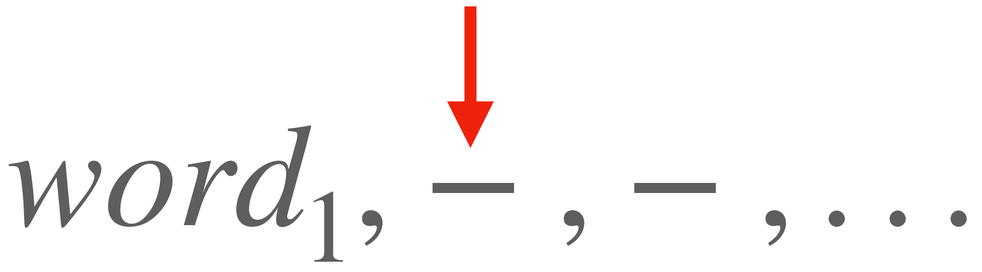
$$-, word_2, word_3, \ldots, word_n$$

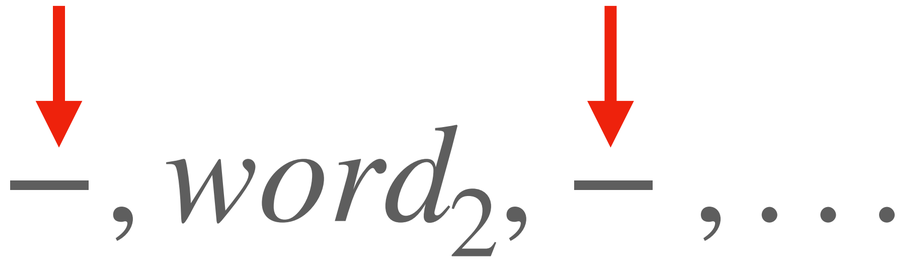

—, word2, word3, . . . , wordn


$word_1, -, word_3, \ldots, word_n$


$word_1, word_2, \_, \ldots, word_n$

Continuous Skip-gram Model


$word_1, \underline{\ } , \underline{\ } , \ldots$



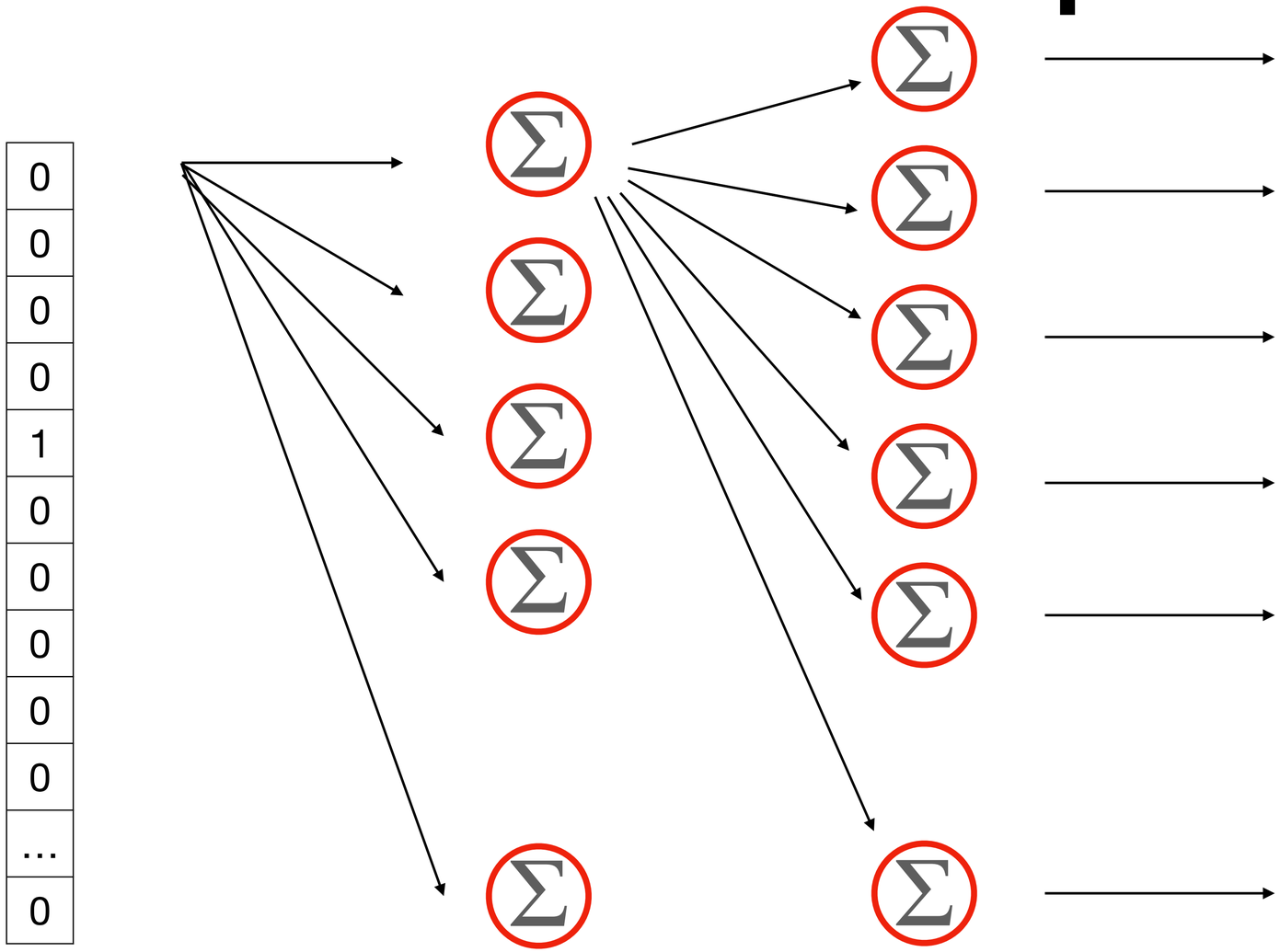
Training Objective

- The objective of Word2Vec is to maximize the likelihood of predicting the context words or target word given the input words.
- For CBOW, the model maximizes the average log probability of the target word given the context words.
- For Skip-gram, the model maximizes the average log probability of the context words given the target word.

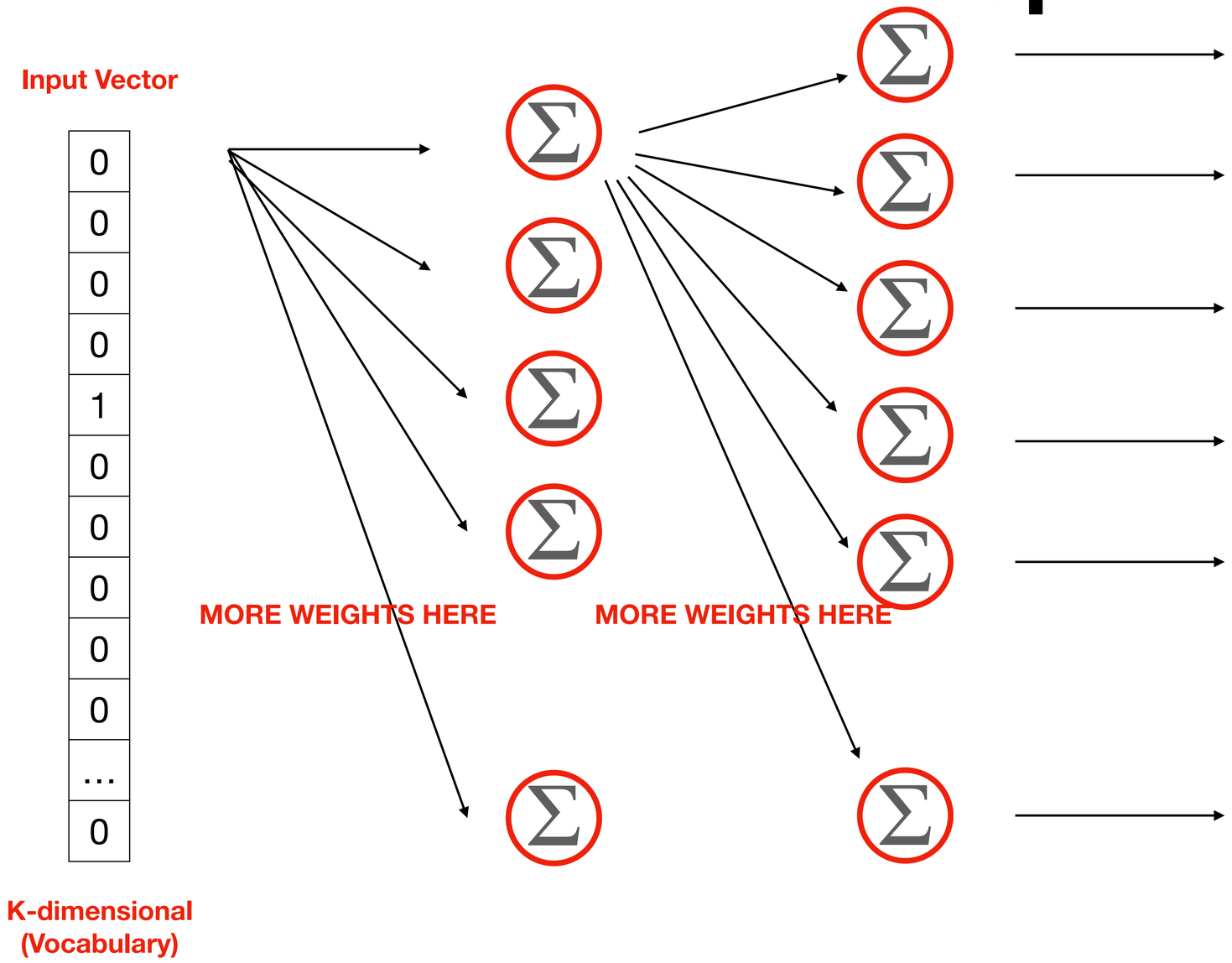
Training Process

- during training, a sliding window is used to define the context words around each target word.
- the words are represented as one-hot encoded
- stochastic gradient descent
- the weights of the ***hidden layer***, which represent the word embeddings, are learned by adjusting the model's parameters to improve the prediction accuracy.

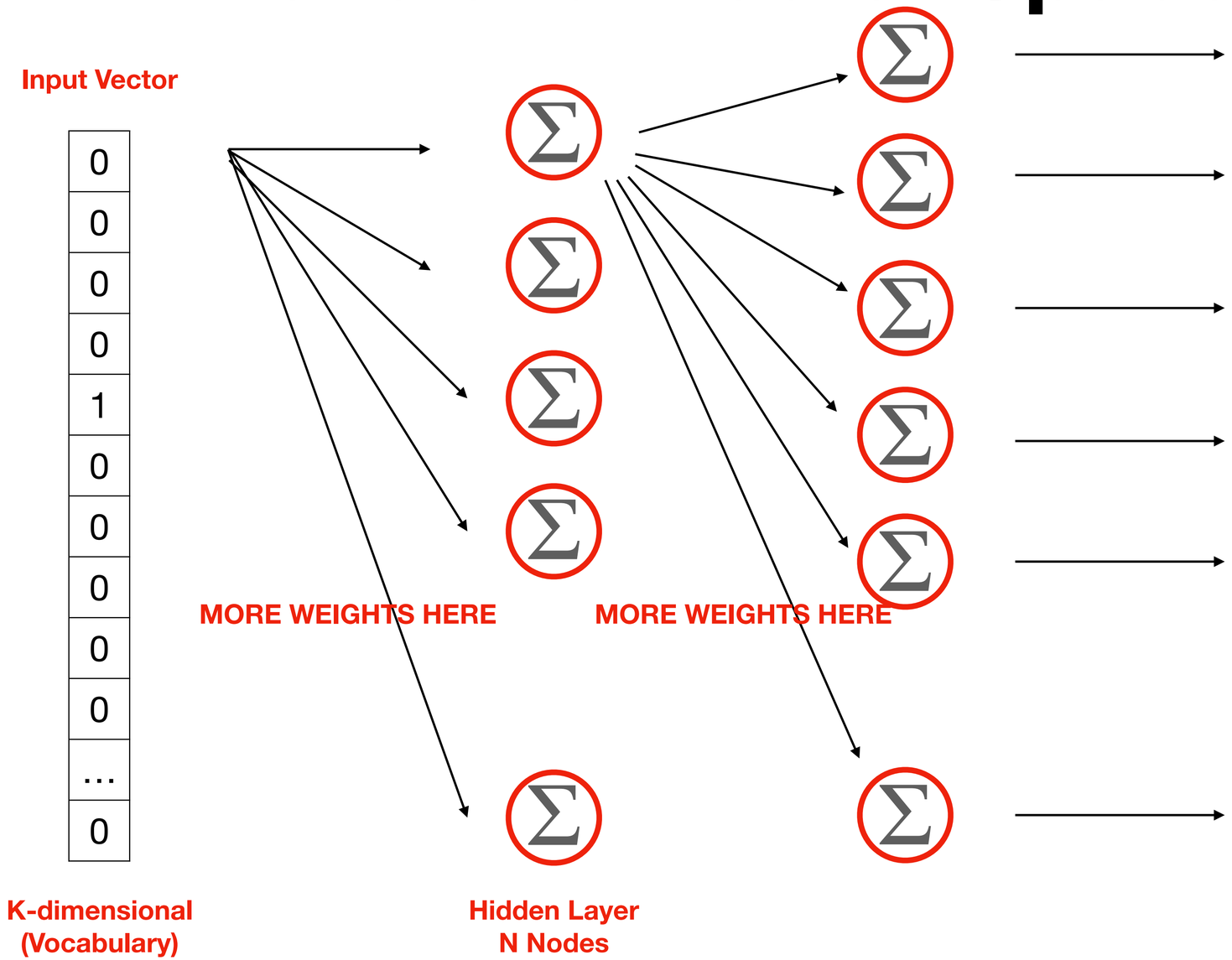
N-Dimensional Vector Space



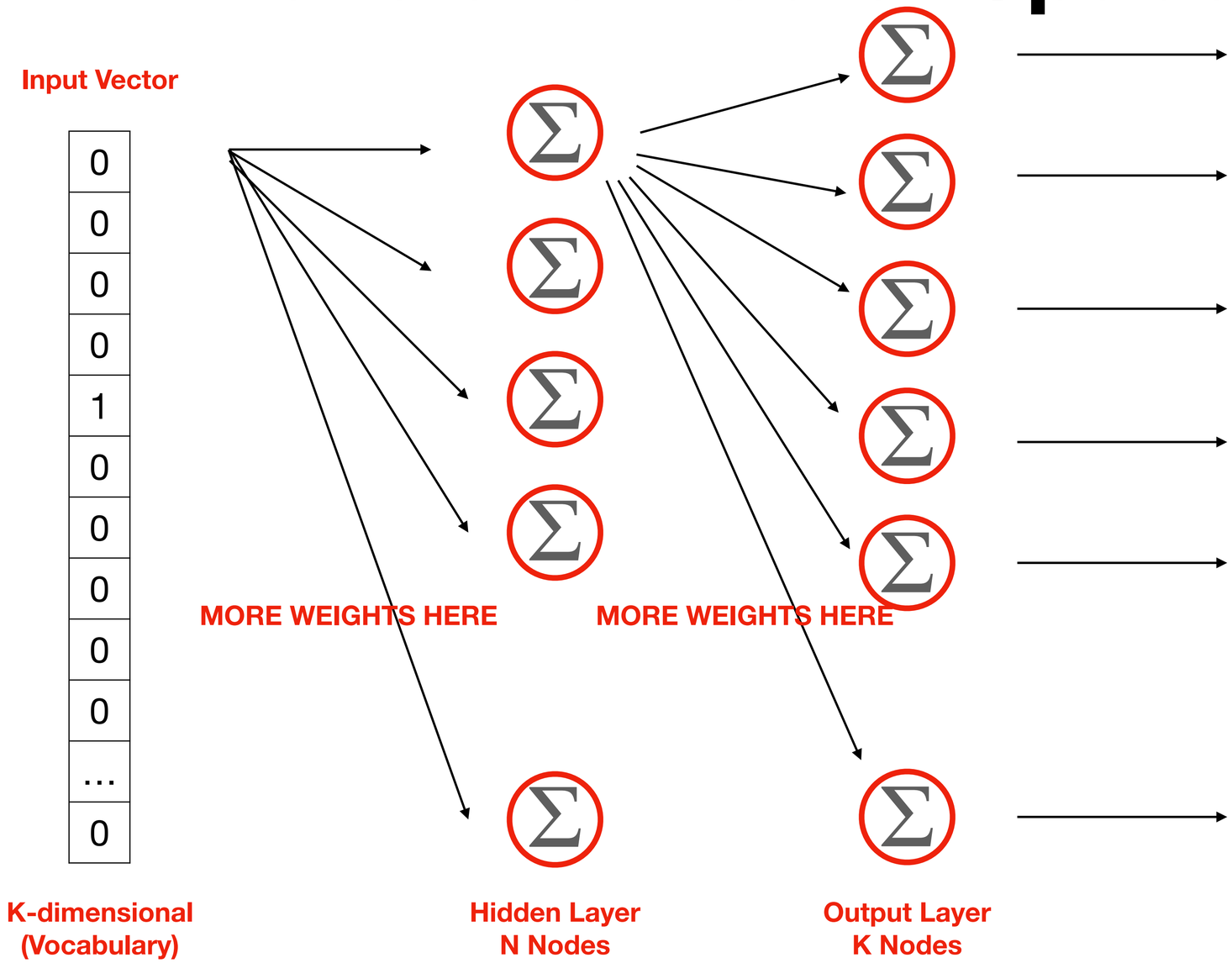
N-Dimensional Vector Space



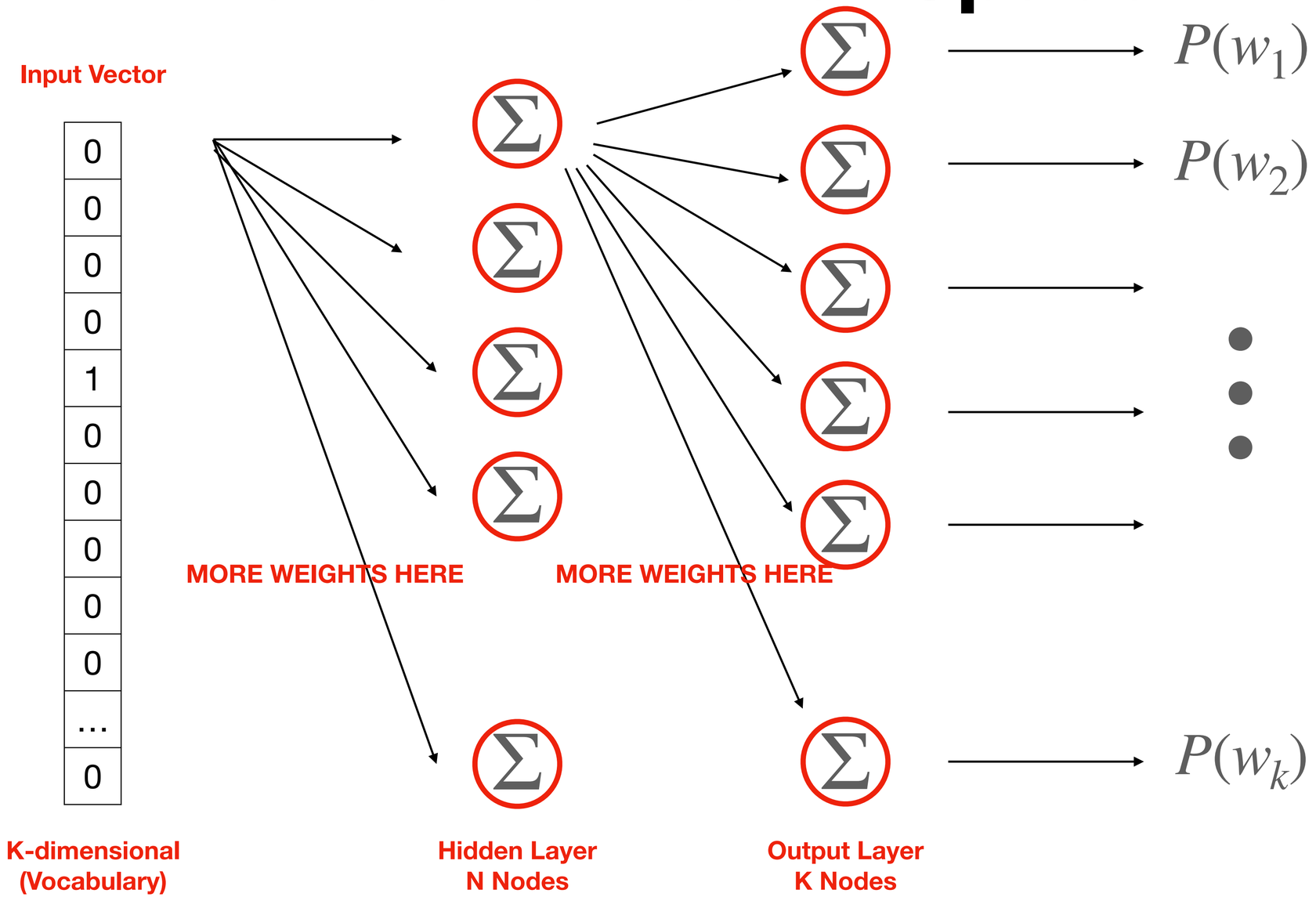
N-Dimensional Vector Space



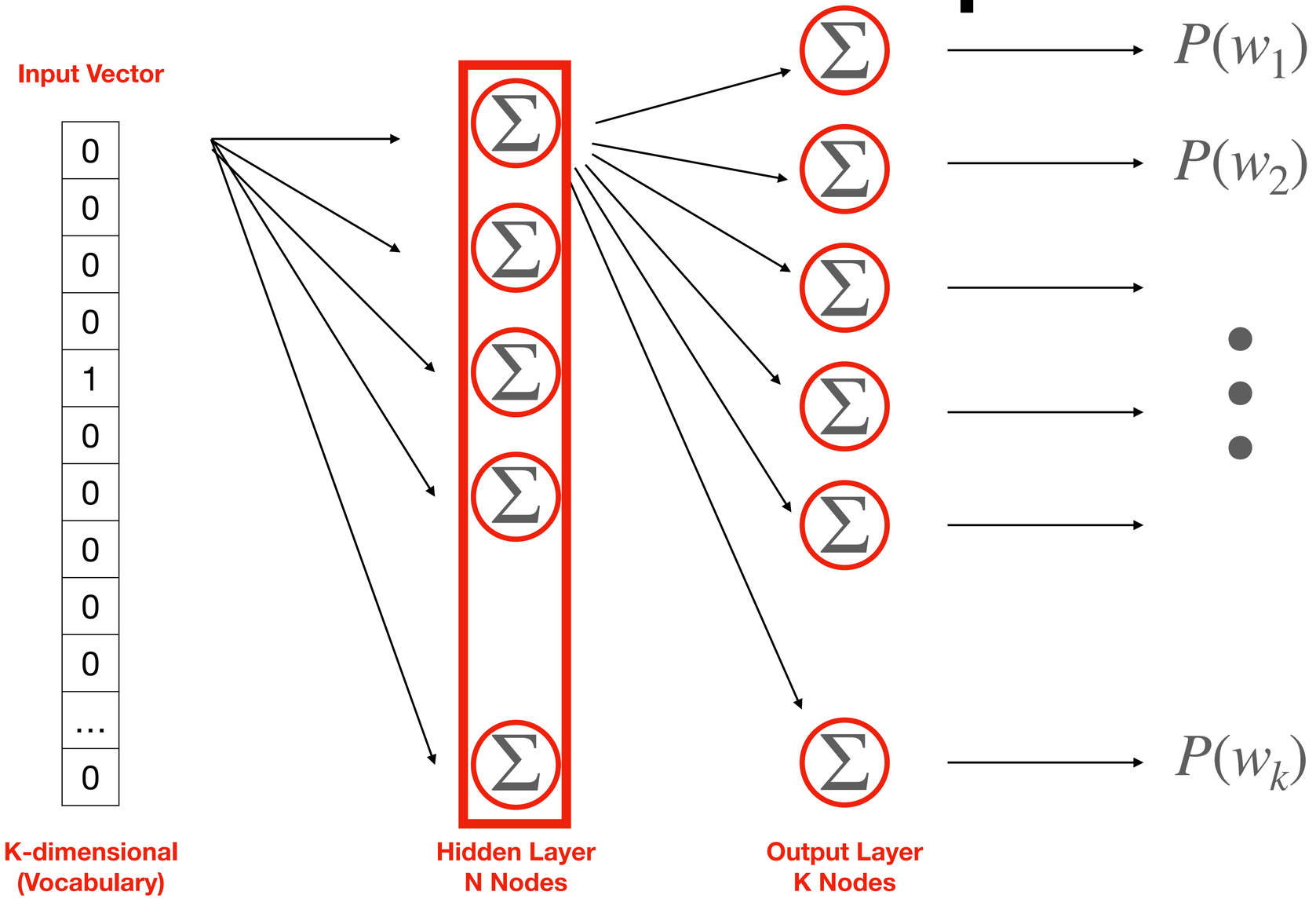
N-Dimensional Vector Space



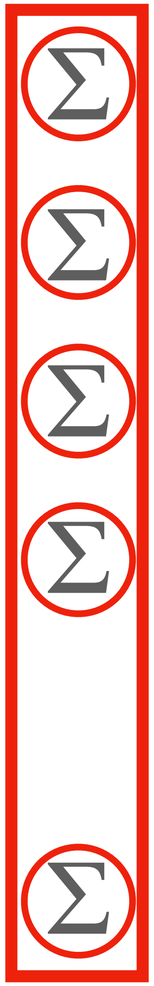
N-Dimensional Vector Space



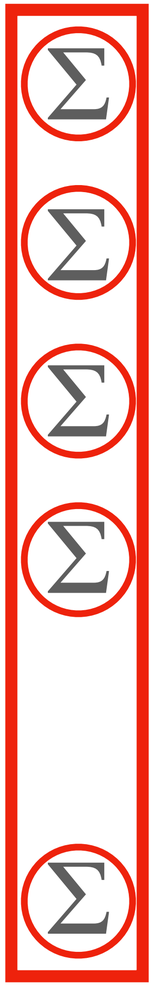
N-Dimensional Vector Space



N-Dimensional Vector Space



N-Dimensional Vector Space



N-Dimensional Vector Space

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

GloVe

- Global Vectors for Word Representation
- focuses on learning word embeddings by leveraging co-occurrence statistics and factorizing a word co-occurrence matrix

GloVe

- co-occurrence matrix that captures the frequency of word co-occurrences in a large corpus
- each entry in the co-occurrence matrix represents the number of times two words co-occur within a specified context window
- size of the context window determines the range of words considered as context words for each target word

Objective Function

- custom objective function
- designed to capture the ratio of co-occurrence probabilities of words
- the aim is to find word embeddings that preserve semantic relationships based on the observed co-occurrence statistics

Contextualized word embeddings

- captures the contextual meaning of words within a given sentence or text
- pretrained on a large corpus of text using a masked language modeling objective and next sentence prediction objective
- during MLM, random words in the input text are masked
- NSP training involves predicting whether two sentences appear consecutively or not, enabling the model to understand relationships between sentences

Transformer Architecture

- In a bit…

Tokenization

- tokenizes input text into subword units (WordPiece or SentencePiece)
- subword token is associated with an embedding vector, and the embeddings of multiple subwords form the representation for a given word

Contextualized Word Representations

- entire sentence or text is fed into the model
- compute contextualized representations for each subword token based on its surrounding context in the sentence
- contextual meaning

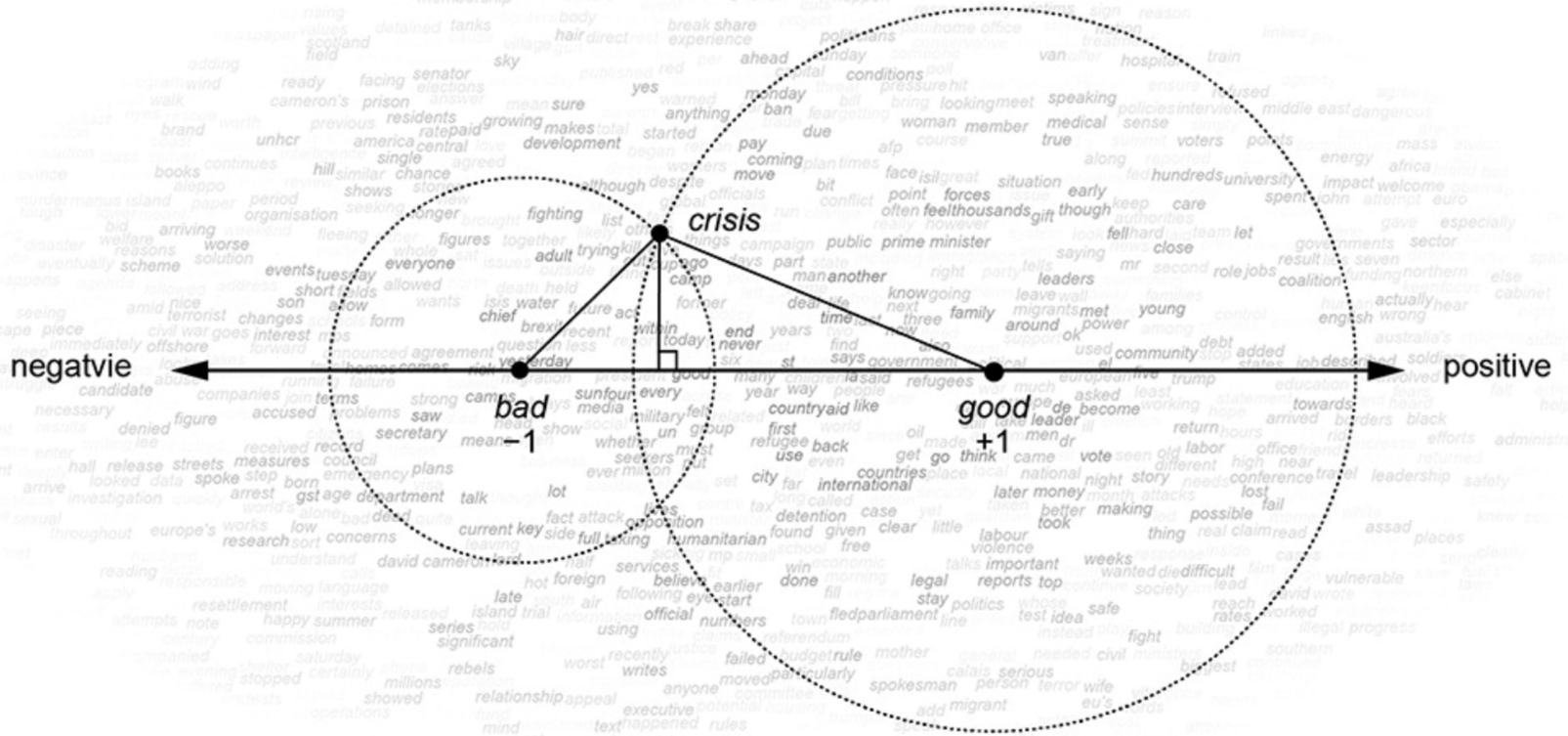
Limitations

- Semantic Shifts
- Difficulty in Handling Out-of-Vocabulary Words
- Lack of Interpretability
- Limited Handling of Syntax and Word Order
- Polysemy and Homonymy

Latent Semantic Scaling

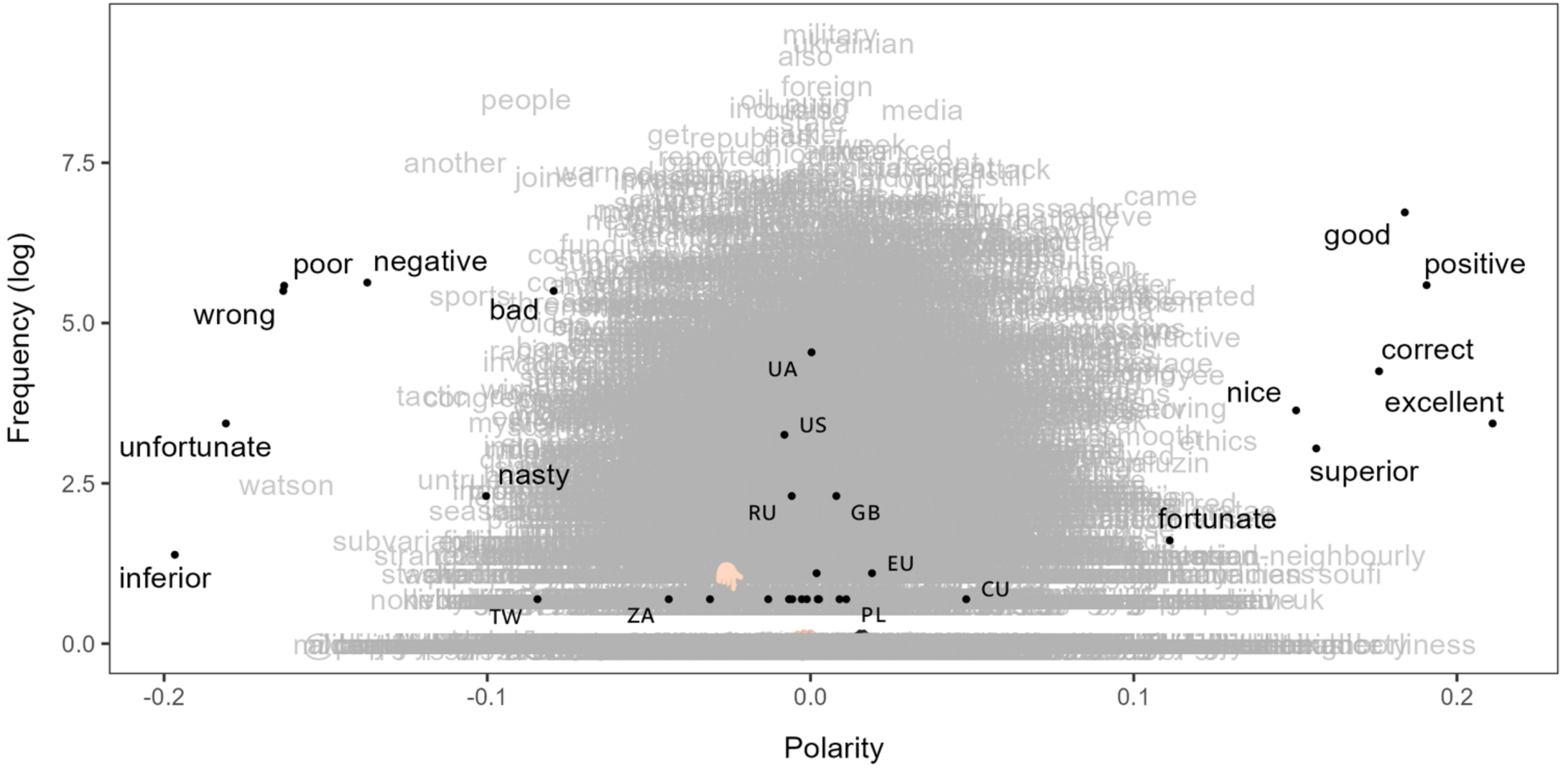
- (e.g., Watanabe, 2020)

Latent Semantic Scaling



Latent Semantic Scaling

- choose “seed words” that represent your dimensions
- calculate the polarity of words based on their proximity to seed words
    - e.g., cosine similarity
- predict polarity scores of documents by weighting word polarity scores by their frequency in the documents


Frequency (log)
7.5
5.0
2.5
0.0
-0.2
-0.1
0.0
0.1
0.2
Polarity
poor
negative
wrong
bad
unfortunate
nasty
inferior
good
positive
correct
nice
excellent
superior
fortunate
UA
US
RU
GB
EU
CU
TW
ZA
PL
people
another
military
ukrainian
also
foreign
media
watson

AMERICAN SOCIOLOGICAL ASSOCIATION

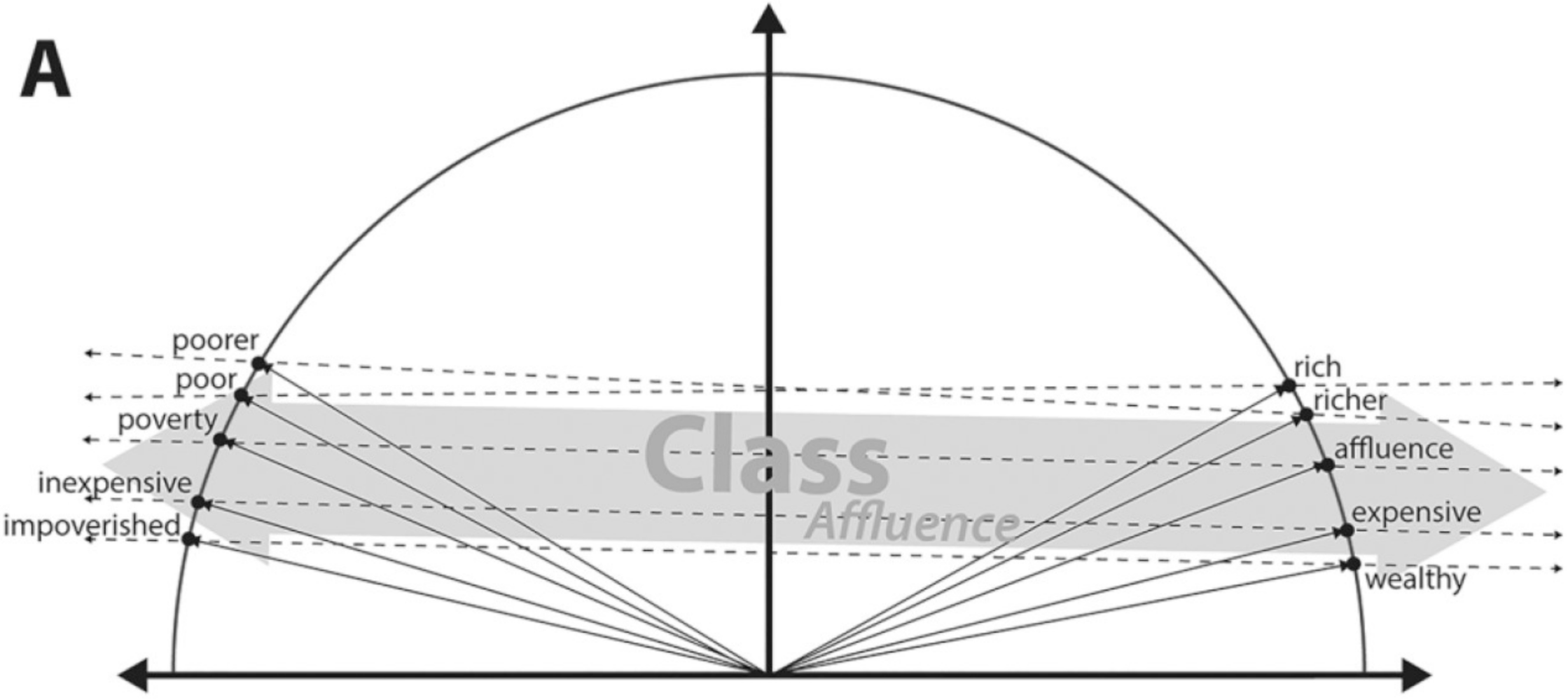
American Sociological Review
2019, Vol. 84(5) 905–949

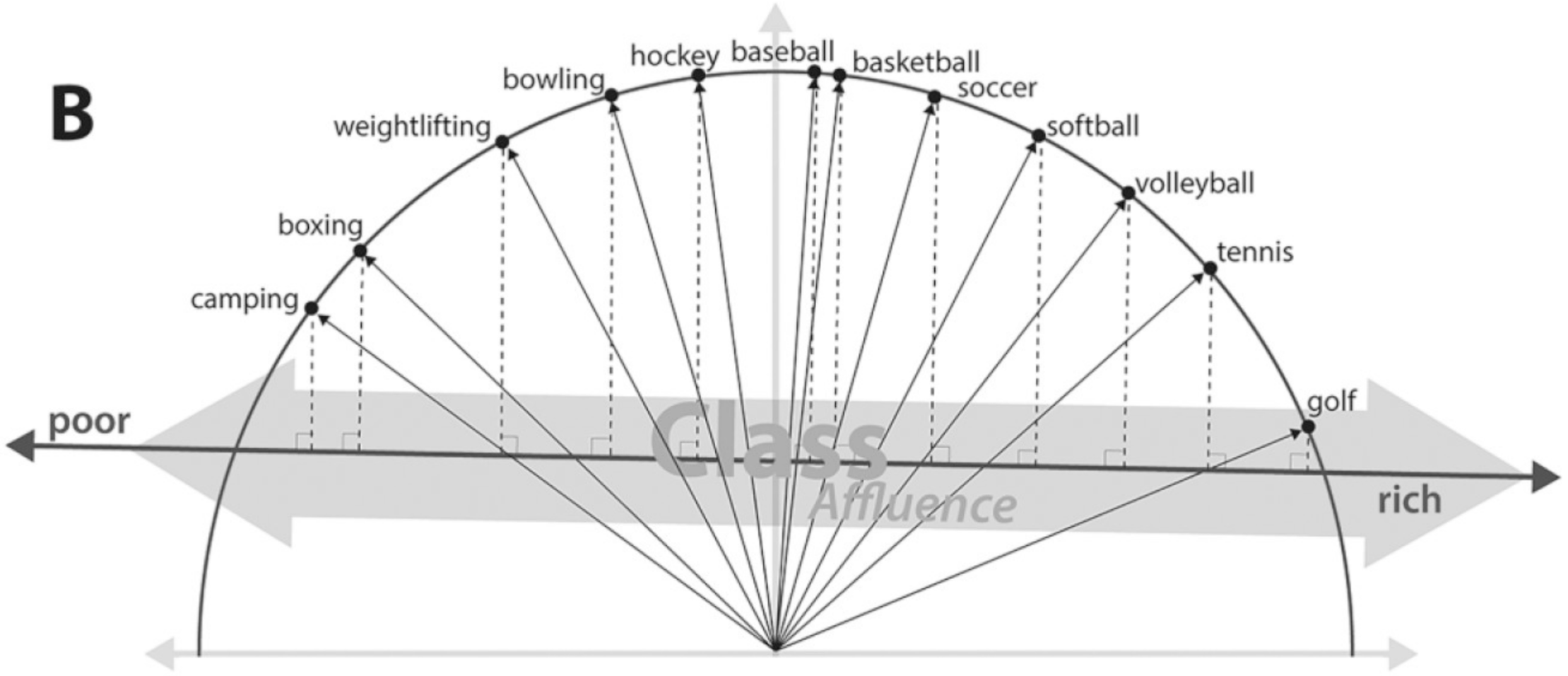
DOI: 10.1177/0003122419877135
journals.sagepub.com/home/asr

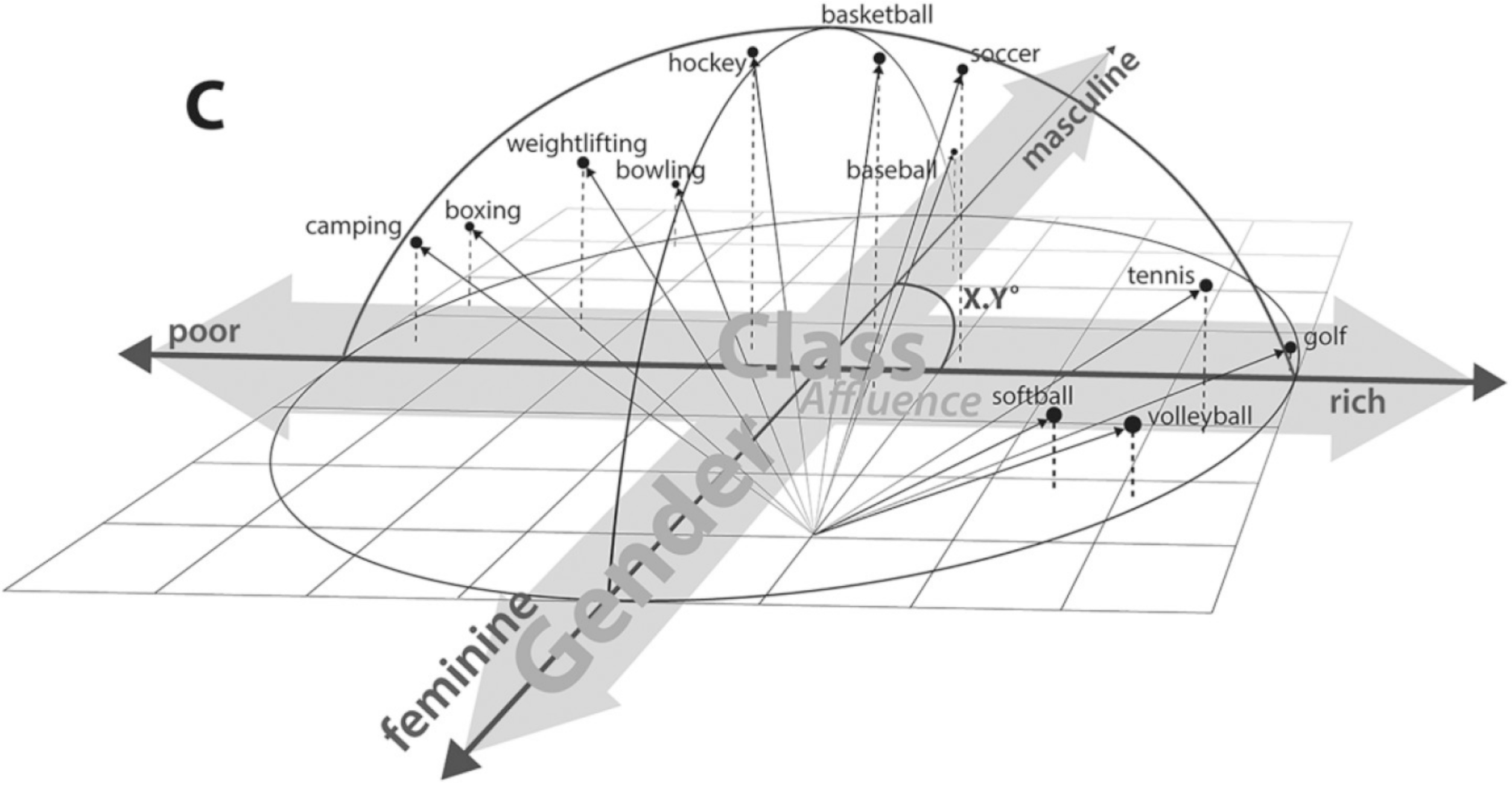
SAGE

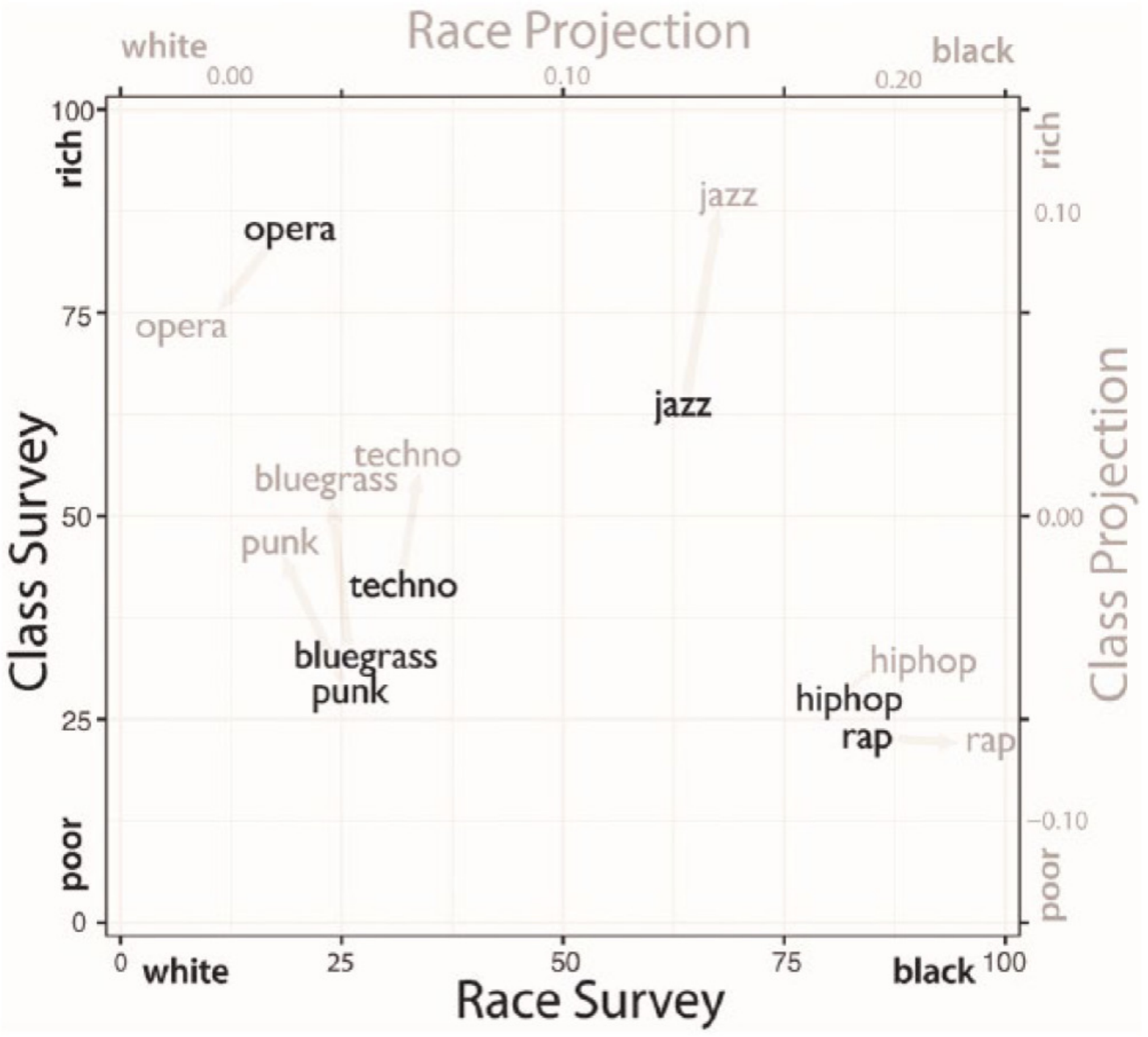
The Geometry of Culture: Analyzing the Meanings of Class through Word Embeddings

Austin C. Kozlowski,[a] Matt Taddy,[b] and James A. Evans[a,c]


A
poorer
poor
poverty
inexpensive
impoverished
rich
richer
affluence
expensive
wealthy
Class
Affluence


B
hockey
baseball
basketball
bowling
soccer
weightlifting
softball
volleyball
boxing
tennis
camping
golf
poor
Class
Affluence
rich


C
basketball
hockey
soccer
masculine
weightlifting
bowling
baseball
camping
boxing
tennis
X.Y°
poor
Class
Affluence
golf
softball
volleyball
rich
Gender
feminine


Race Projection
white
black
0.00
0.10
0.20
rich
poor
Class Survey
100
75
50
25
0
opera
opera
jazz
jazz
techno
bluegrass
punk
techno
bluegrass
punk
hiphop
hiphop
rap
rap
0
white
25
50
75
black
100
Race Survey
rich
poor
Class Projection
0.10
0.00
−0.10

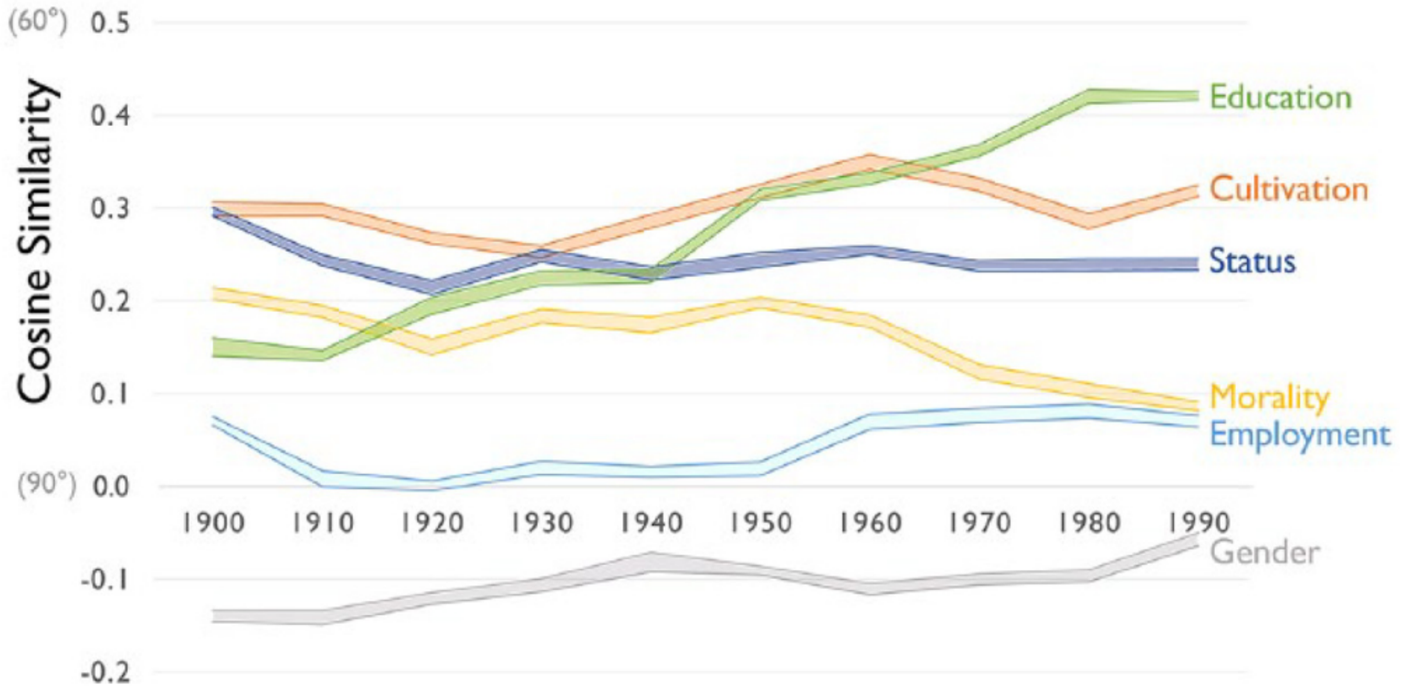


Figure 5. Cosine Similarity between the Affluence Dimension and Six Other Cultural Dimensions of Class by Decade; 1900 to 1999 Google Ngrams Corpus

Note: Bands represent 90 percent bootstrapped confidence intervals produced by subsampling.

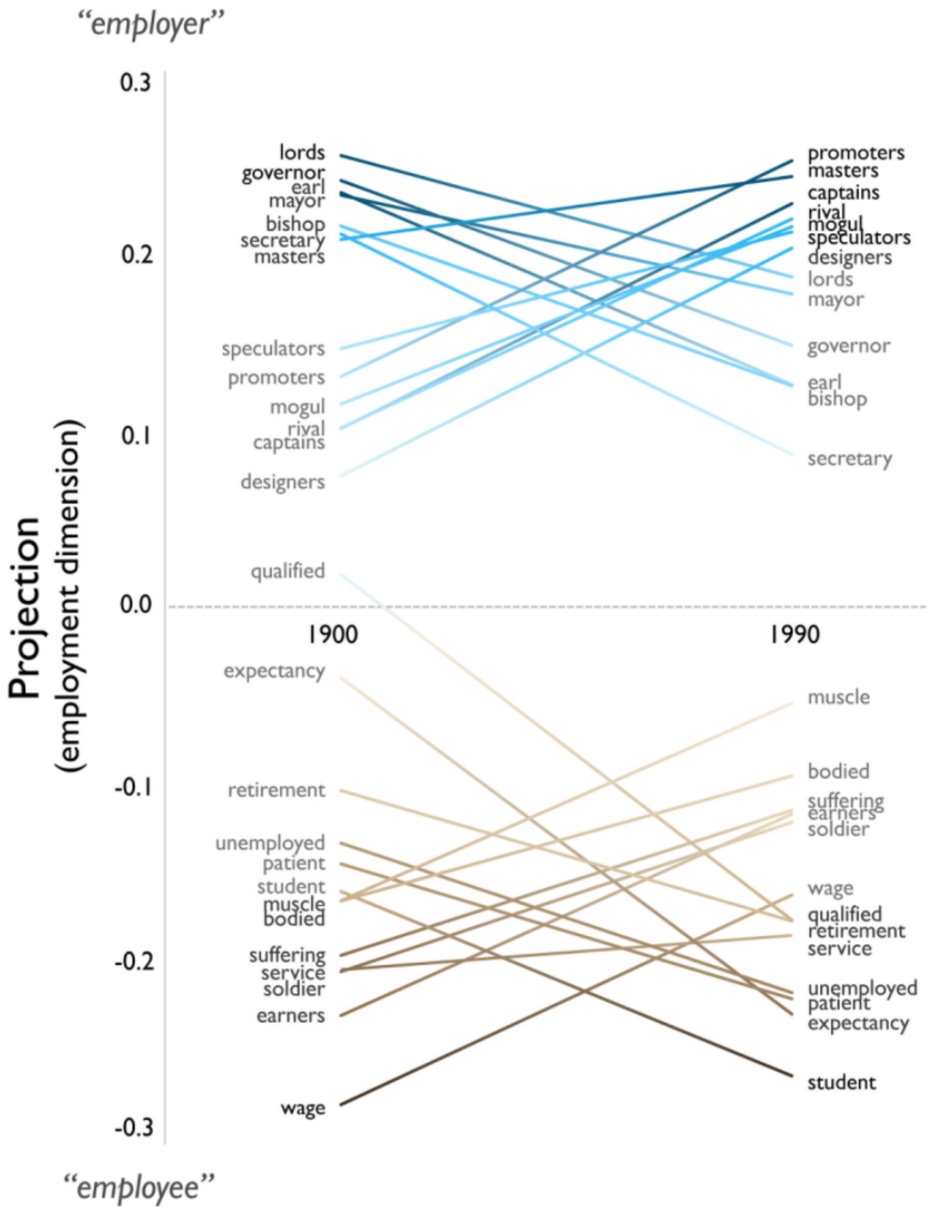


Figure 10. Words That Project High and Low on the Employment Dimension of Word Embedding Models Trained on Texts Published at the Beginning and End of the Twentieth Century; 1900–1919 and 1980–1999 Google Ngrams Corpus

Transformer Models

Language Model

- probability distribution over a sequence of words
- e.g., Markov Chains language model

Large Language Model

- probability distribution over a sequence of words
- Trained on the large chunk of the internet
- ~100TB of text data
- (Rumours) GPT-4 cost ~$200m to train
- Next-word prediction task

Large Language Model

- probability distribution over a sequence of words
- Trained on the large chunk of the internet
- ~100TB of text data
- (Rumours) GPT-4 cost ~$200m to train
- ***Next-word prediction task***

Large Language Model

- probability distribution over a sequence of words
- Trained on the large chunk of the internet
- ~100TB of text data
- (Rumours) GPT-4 cost ~$200m to train
- ***Next-word prediction task***
- ***Requires a lot of background “knowledge”***

Transformer Models

- A lot more complicated than what we've seen…

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Attention is all you need

- 105k citations

Attention is all you need

- 105k citations
- In ~ 5 years

Attention is all you need

- 105k citations
- In ~ 5 years
- Developed for machine translation
- Blows all previous NN architectures (like RNNs, LSTMs, etc.) out of the water

Attention is all you need

- 105k citations
- In ~ 5 years
- Developed for machine translation
- Blows all previous NN architectures (like RNNs, LSTMs, etc.) out of the water
- Bidirectional Encoder Representation from Transformers
- Generative Pre-trained Transformer

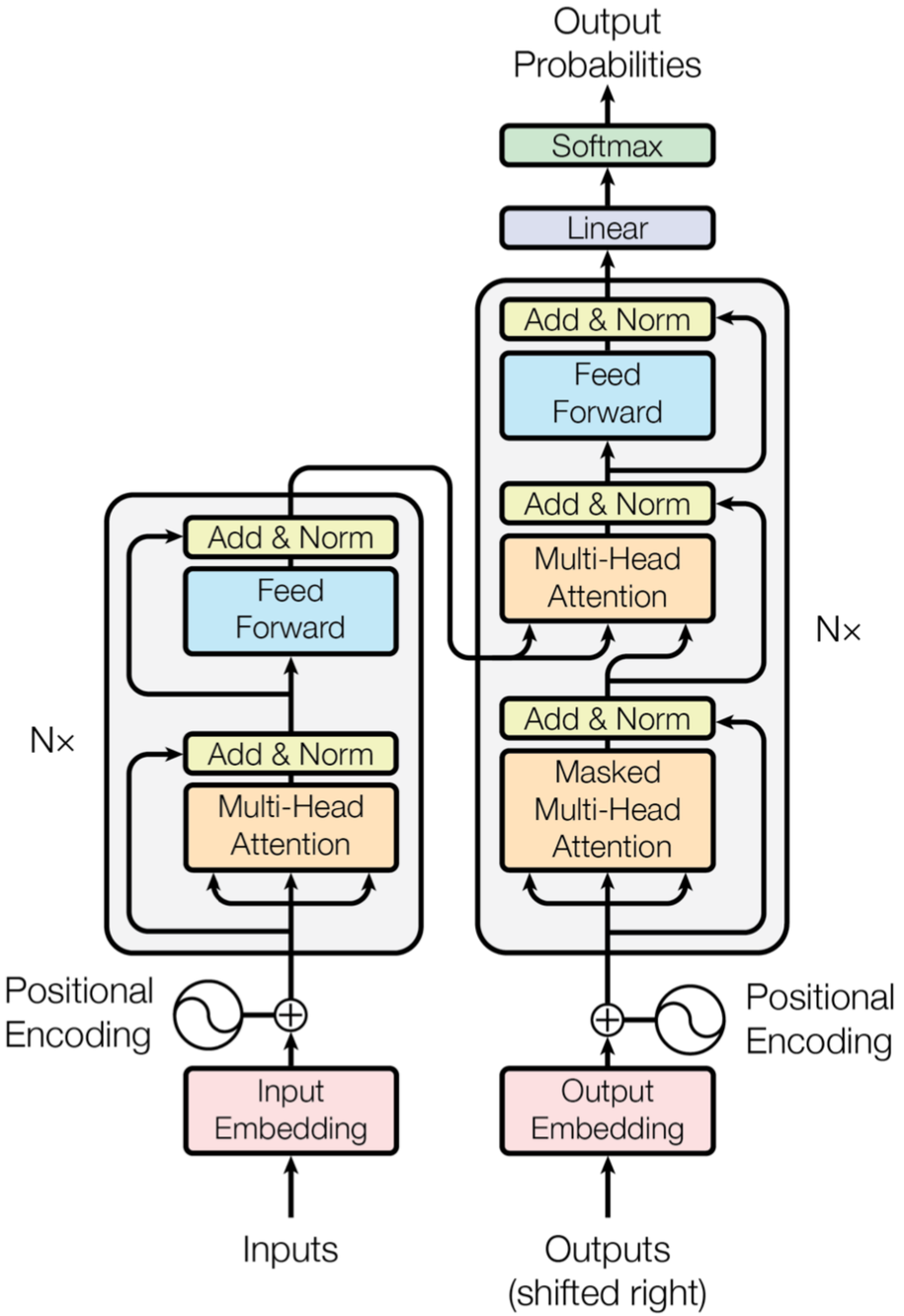
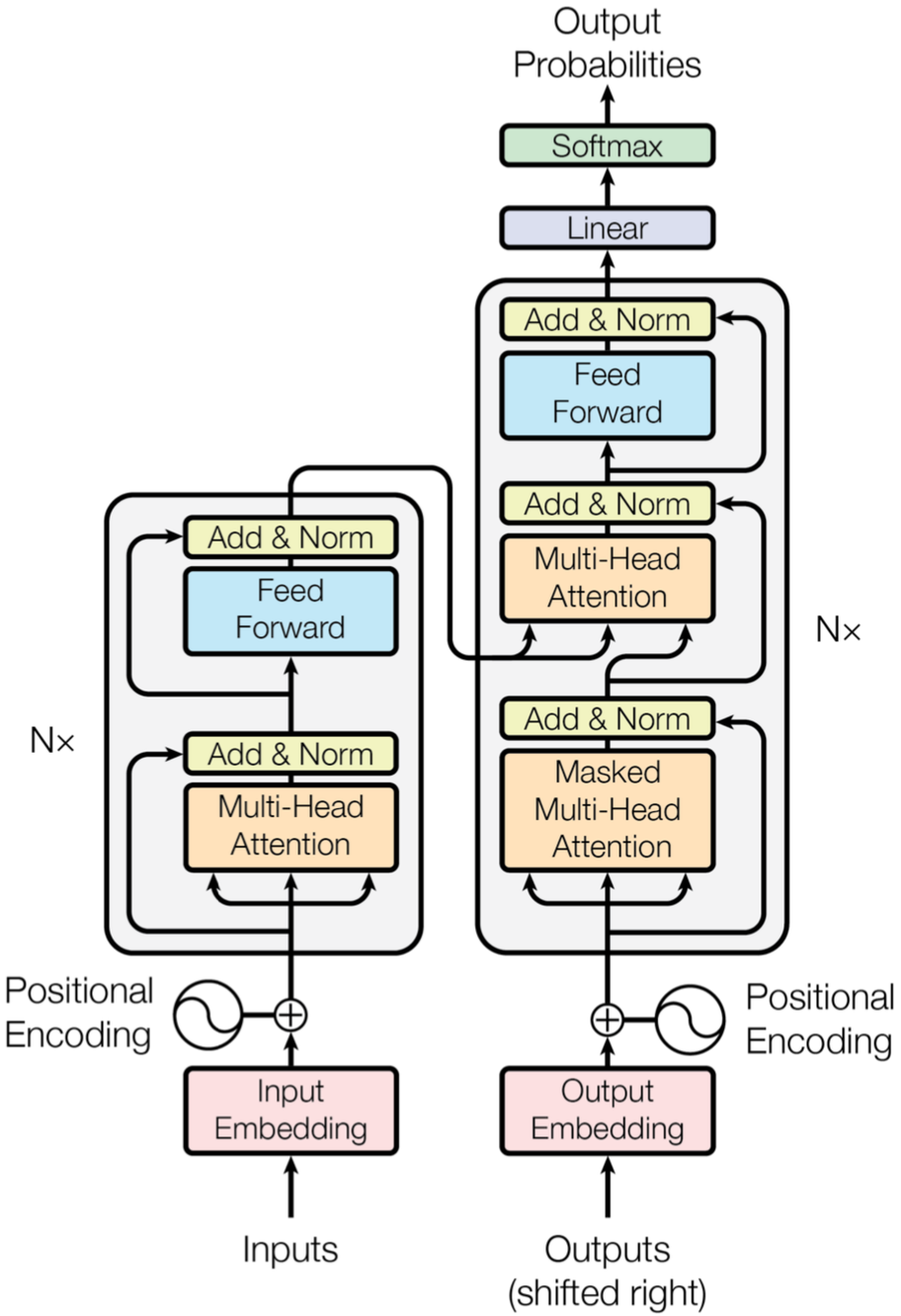


Figure 1: The Transformer - model architecture.

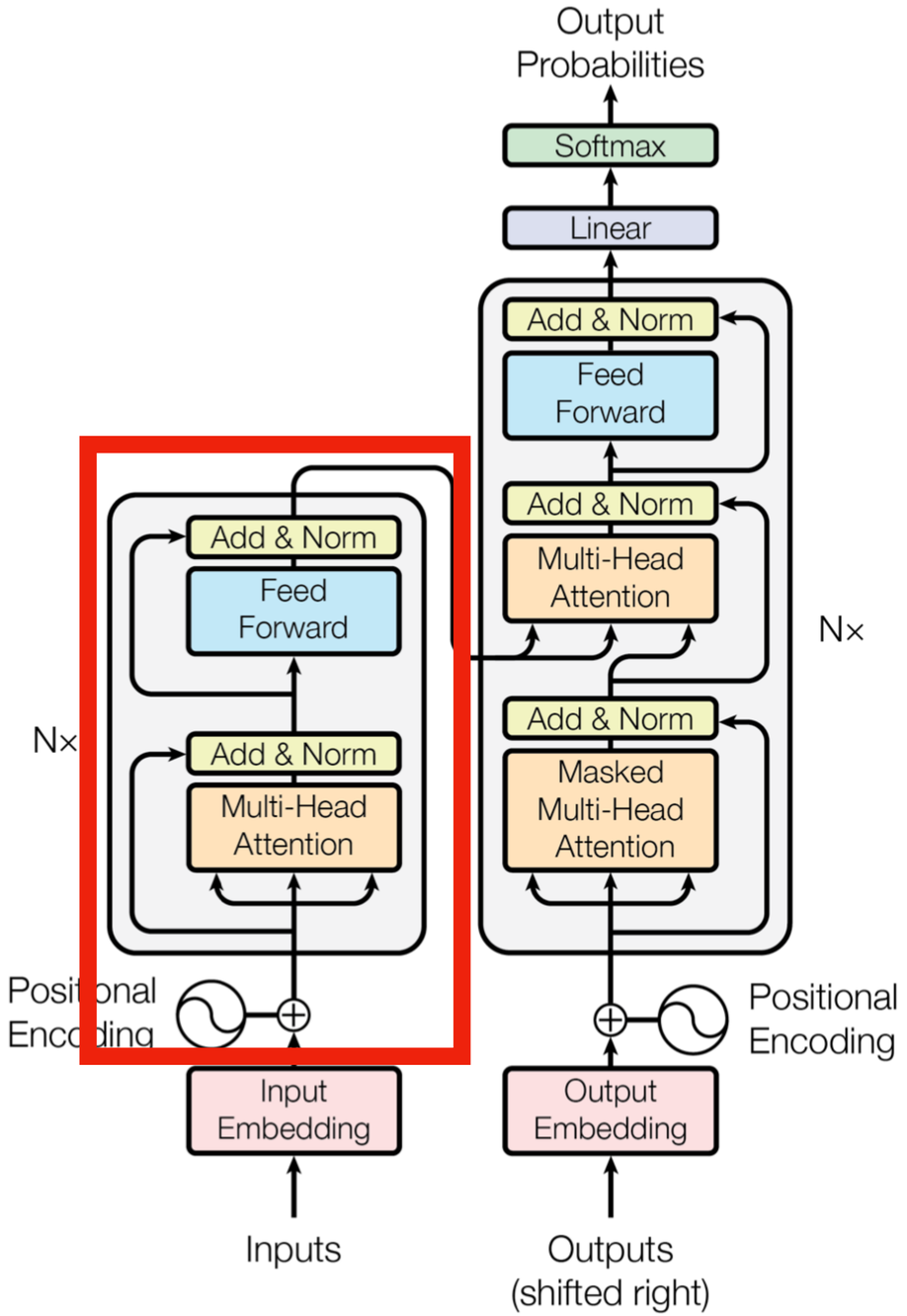
Encoder



Decoder

Figure 1: The Transformer - model architecture.

Encoder



Decoder

Figure 1: The Transformer - model architecture.

Positional Encodings

I love ice-cream

Positional Encodings

I love ice-cream

1 2 3

Self-Attention module

- Handles long-term dependencies
- Pays “attention” to which parts of the input sequence are important
- The self-attention mechanism allows each position in the input sequence to attend to all other positions, weighing the importance of different positions during the computation
- capture long-range dependencies and relationships between words or tokens in the sequence effectively

Encoding for self-attention

- Query
- Key
- Value

Encoding for self-attention

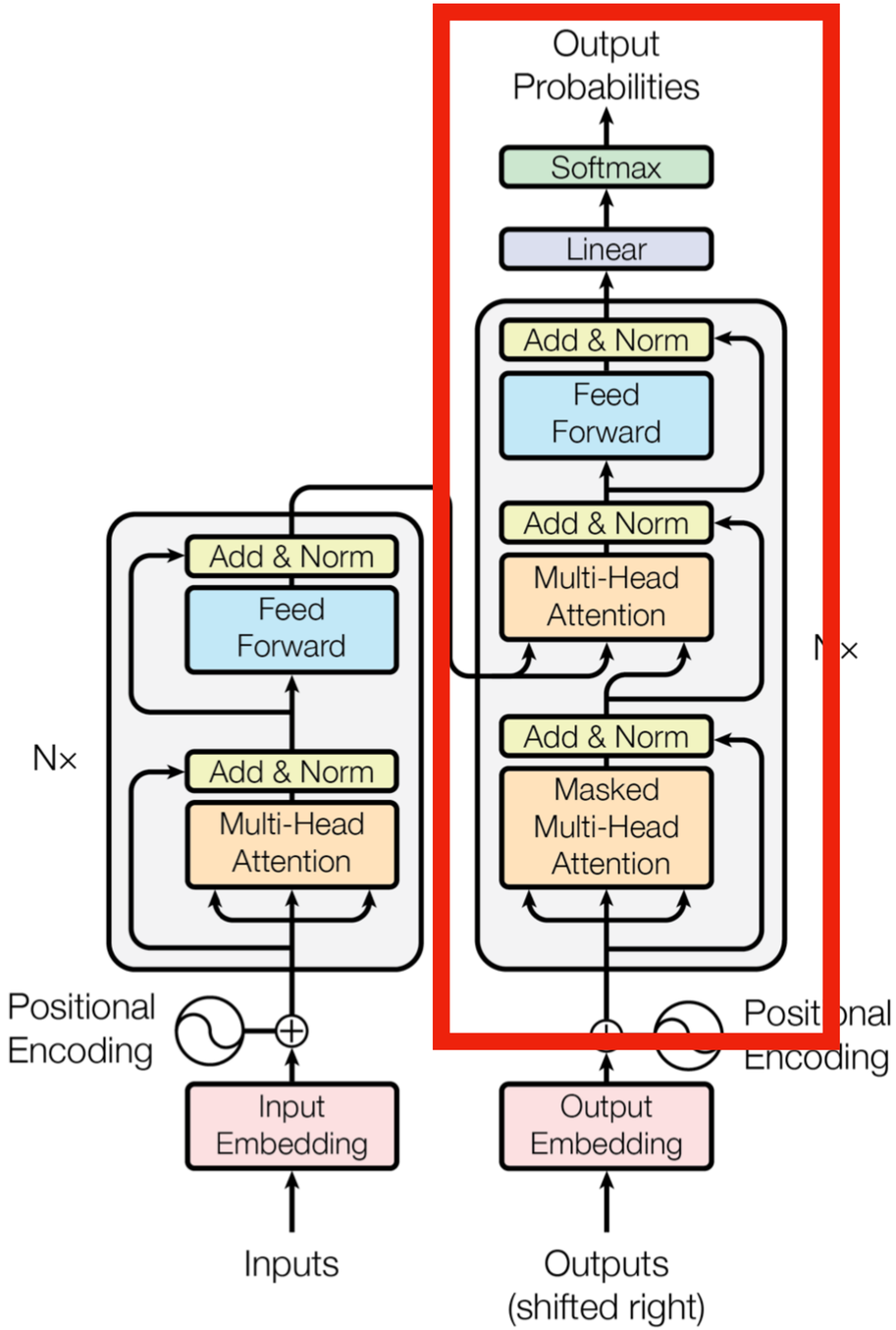
- For each token in the sequence, the model calculates query, key, and value vectors by applying learned linear transformations to the token embeddings
- The model computes the attention weights by measuring the similarity between the query vector of a token and the key vectors of all tokens in the sequence
- The attention weights obtained in the previous step are used to weight the value vectors of each token
- The resulting weighted sum is the transformed representation of the original token

	I	Love	Ice-Cream
I	.76	.89	.54
Love	.89	.46	.88
Ice-Cream	.54	.88	.54

	I	Love	Ice-Cream
I	.76	.89	.54
Love	.89	.46	.88
Ice-Cream	.54	.88	.54

Value

Encoder



Decoder

Figure 1: The Transformer - model architecture.

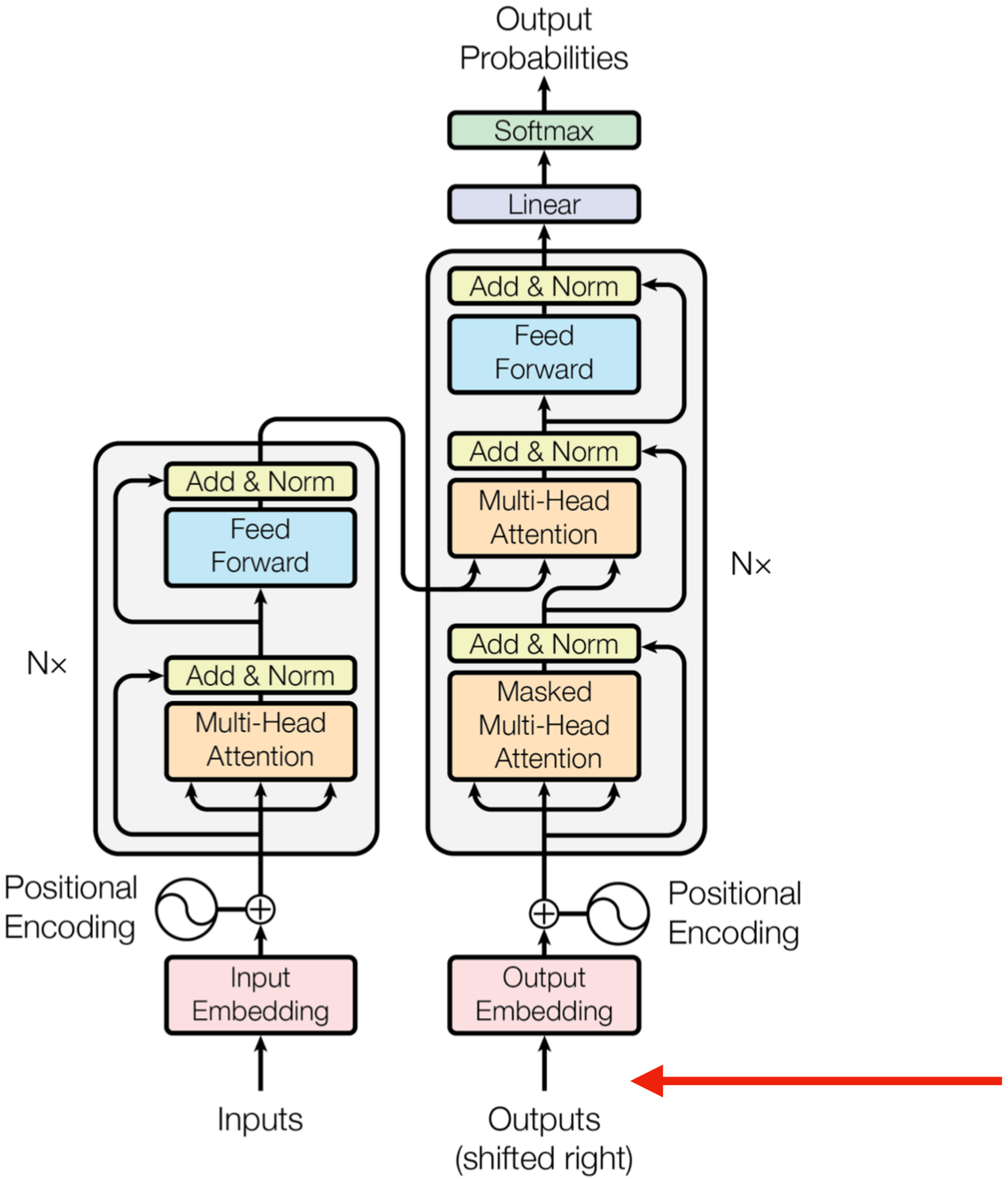


Figure 1: The Transformer - model architecture.

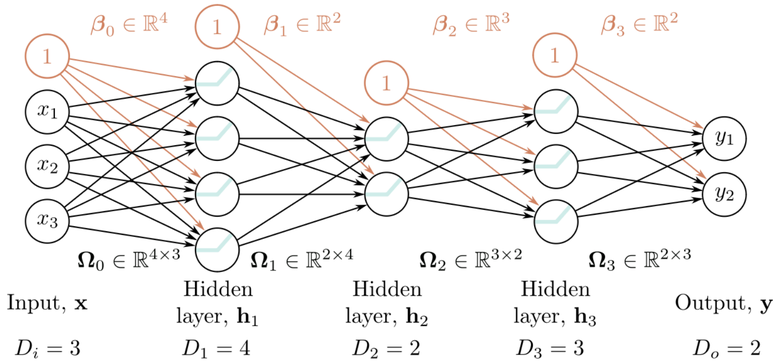
Why “Transformers”?

Why “Transformers”?

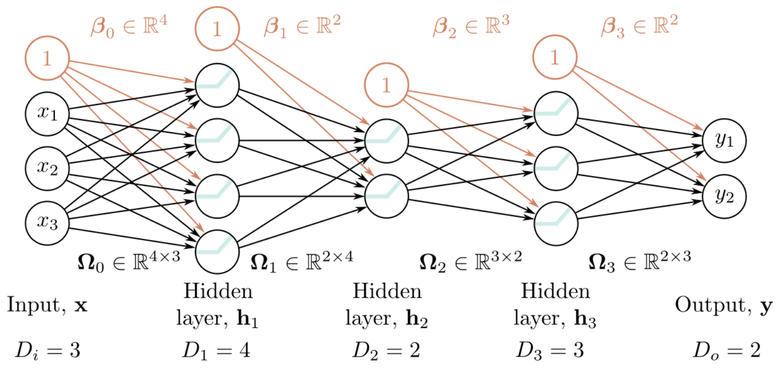
- model “transforms” the representation of each input token by attending to different positions in the sequence

Fine Tuning

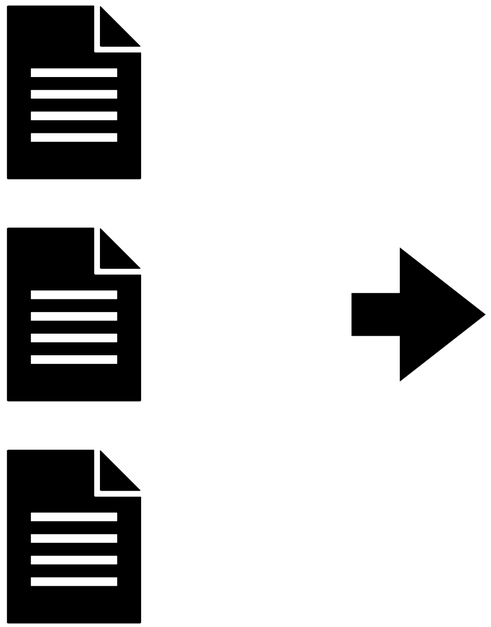
Fine Tuning


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

Fine Tuning


$\boldsymbol{\beta}_0 \in \mathbb{R}^4$
$\boldsymbol{\beta}_1 \in \mathbb{R}^2$
$\boldsymbol{\beta}_2 \in \mathbb{R}^3$
$\boldsymbol{\beta}_3 \in \mathbb{R}^2$
1
x_1
x_2
x_3
y_1
y_2
$\boldsymbol{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\boldsymbol{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\boldsymbol{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\boldsymbol{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
$D_i = 3$
Hidden layer, $\mathbf{h}_1$
$D_1 = 4$
Hidden layer, $\mathbf{h}_2$
$D_2 = 2$
Hidden layer, $\mathbf{h}_3$
$D_3 = 3$
Output, $\mathbf{y}$
$D_o = 2$

Assistants (like Chat-GPT)



Input / Output
pairs

Transformer Models for Social Science

- Reduction of labeled data, performance increase
- Pre-trained Language Models:
    - Pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) are used for text classification tasks.
    - These models learn contextual word representations from large amounts of unlabeled text data and can be fine-tuned for specific classification tasks.

Transformers for “Feature Engineering”

- What features are the most important for (e.g.,) discriminating text types?
- Feed the model several example texts
- It distinguishes features that are important
- Use these features to classify/scale/etc., large quantities of data

Open vs. Closed source

- Ethics
- Performance vs. Reproducibility
- Use cases
    - Do you ***need*** it?

Open Source

- Huggingface:
    - https://huggingface.co/
    - Many pre-trained models and datasets
- spaCy:
    - https://spacy.io/
    - Ecosystem for NLP

Open Source

- Better in Python
- But:
    - spaCyR
    - https://github.com/chainsawriot/grafzahl
        - Huggingface wrapper

Your Thoughts?