Tutorial: Neural Networks & Weighted Automata

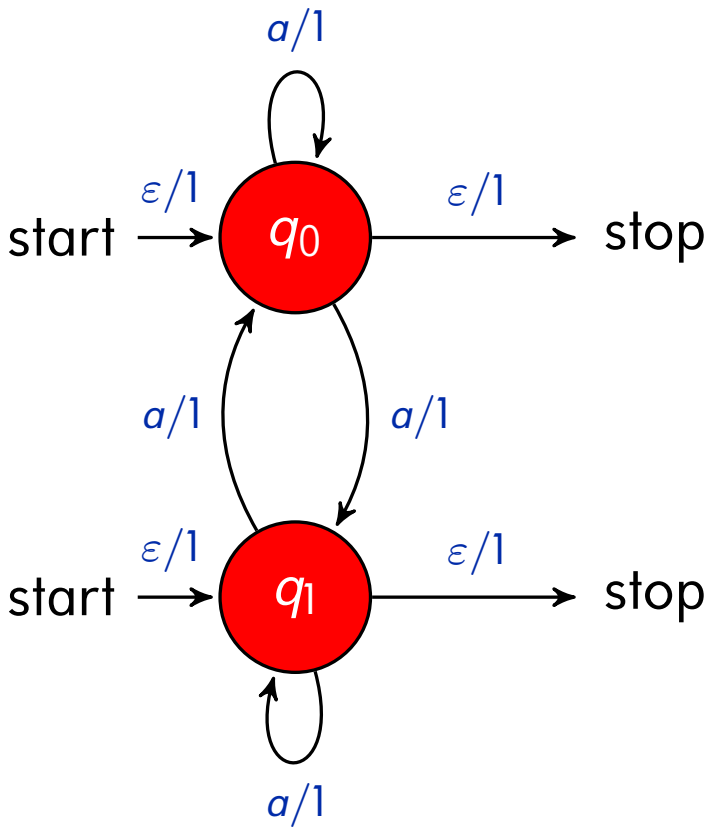
Andreas Maletti

Universität Leipzig, Germany

WATA — April 20, 2021

Weighted automata

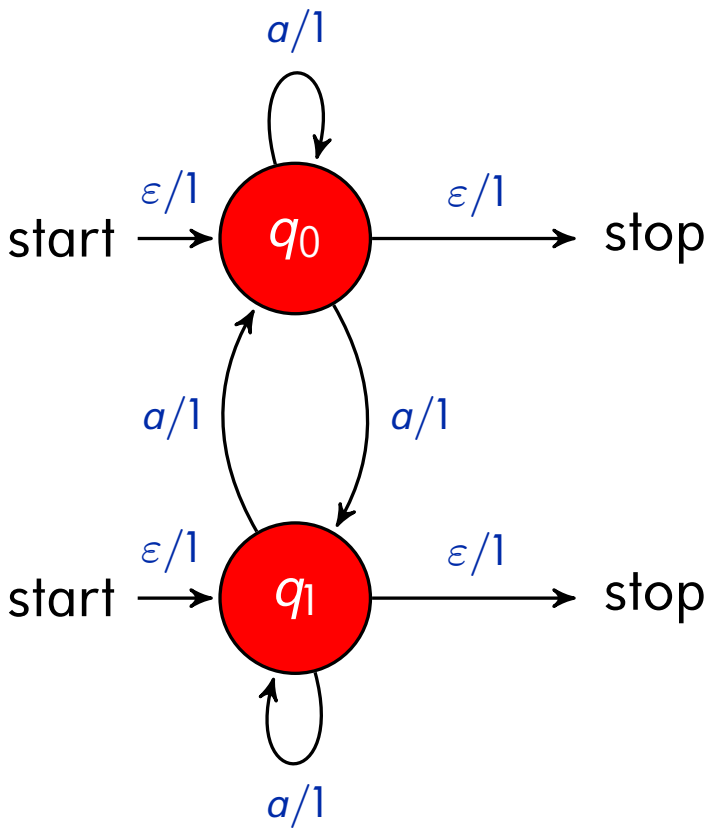
Weight structure: commutative field $(\mathbb{R}, +, \cdot, 0, 1)$



Path from start to stop: Product over labels in $\Sigma^* \times \mathbb{R}$

Weighted automata

- 2^{k+1} paths for input a^k
- Every path has weight 1

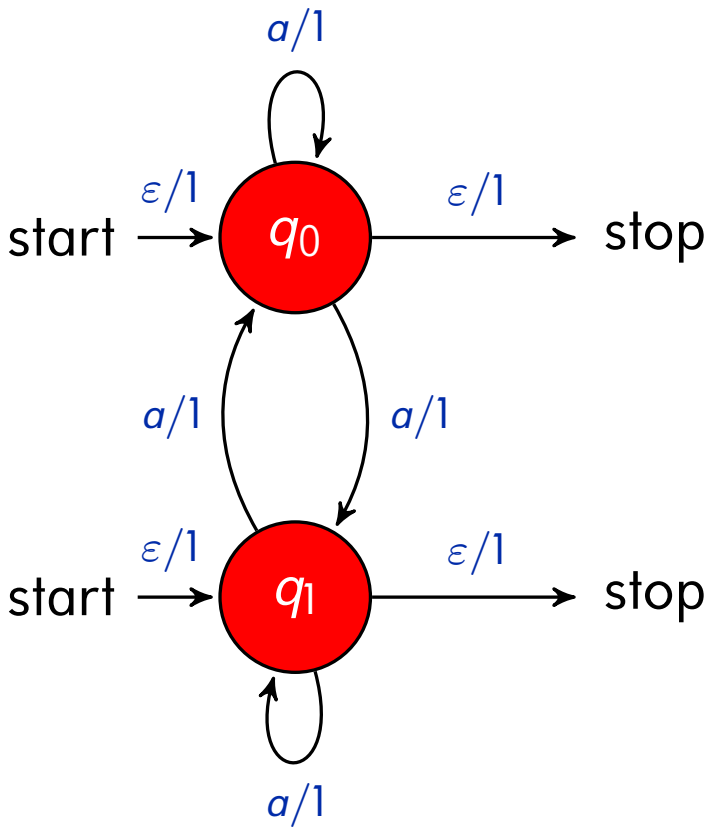


Weighted automata

- 2^{k+1} paths for input a^k
- Every path has weight 1

Weight of input:

$$\mathrm{wt}(a^k) = \sum_{\pi \text{ path for } a^k} \mathrm{wt}(\pi) = 2^{k+1}$$



Weighted automata

Definition (weighted automaton)

Weighted automaton $\mathcal{A} = (Q, \Sigma, I, (M_a)_{a \in \Sigma}, F)$

- finite set Q
- alphabet Σ
- $I \in \mathbb{R}^Q$ initial weight vector
- $M_a \in \mathbb{R}^{Q \times Q}$ transition weight matrix for each $a \in \Sigma$
- $F \in \mathbb{R}^Q$ final weight vector

Weighted automata

Definition (weighted automaton)

Weighted automaton $\mathcal{A} = (Q, \Sigma, I, (M_a)_{a \in \Sigma}, F)$

- finite set Q
- alphabet Σ
- $I \in \mathbb{R}^Q$ initial weight vector
- $M_a \in \mathbb{R}^{Q \times Q}$ transition weight matrix for each $a \in \Sigma$
- $F \in \mathbb{R}^Q$ final weight vector

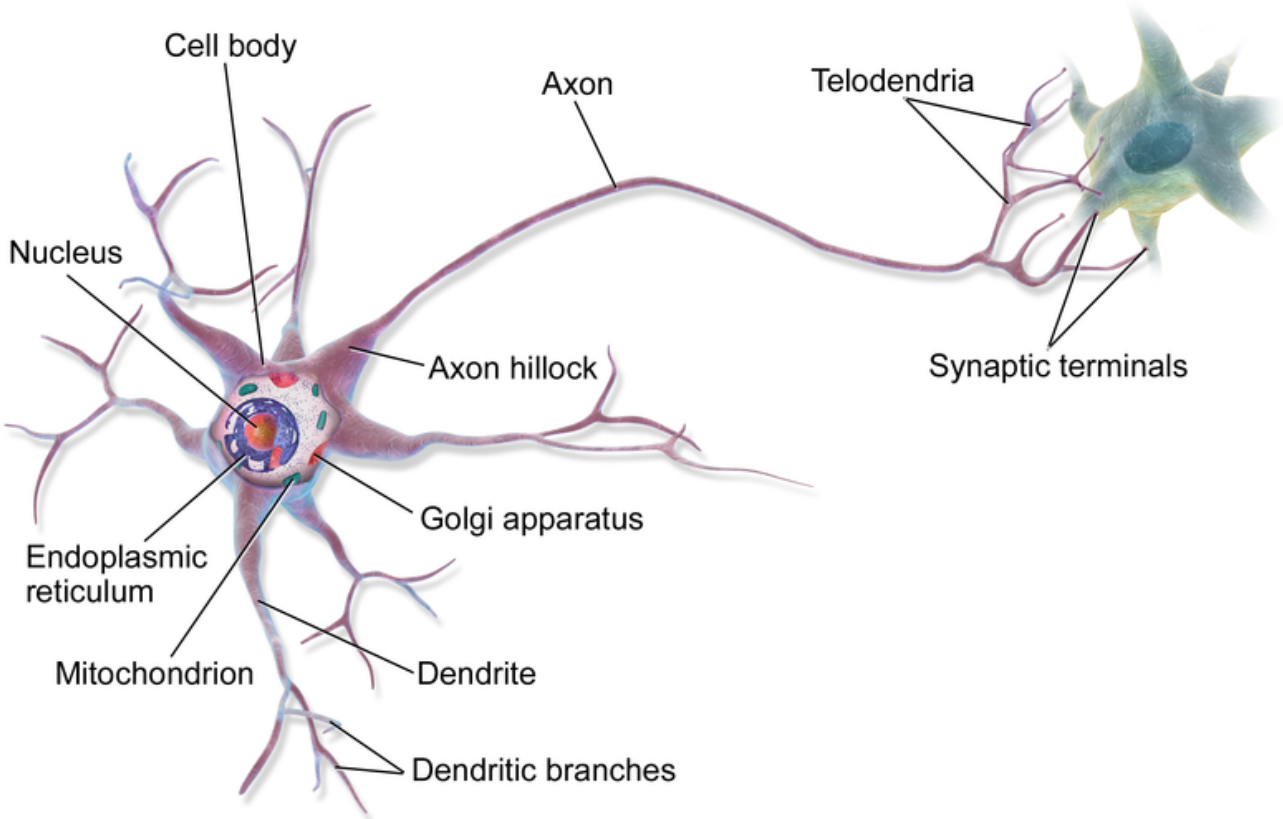
Semantics

$$\mathcal{A}(a_1 \cdots a_k) = I^\mathsf{T} M_{a_1} \cdots M_{a_k} F \qquad \text{for all } a_1, \ldots, a_k \in \Sigma$$

Neural networks

Neural network

- Edge-weighted graph (N, E, wt) of neurons inspired by nerve cells
- Focus on Recurrent Neural Nets (RNN)



©Bruce Blaus

Neural networks

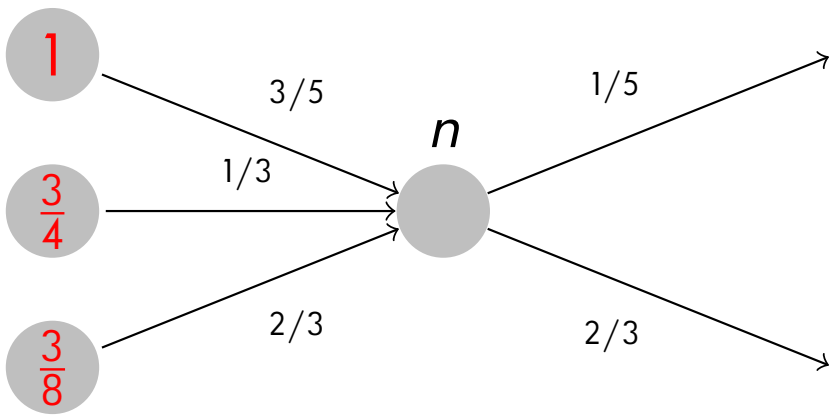
Formalization

- Each neuron $n \in N$ is vertex
- Weight $\mathrm{wt}(n', n) \in \mathbb{R}$ indicates connection strength from n' to n
- Incoming edges of n contribute to activation of n
  (All input potentials moderated by connection strength added up)

Neural networks

Formalization

- Each neuron $n \in N$ is vertex
- Weight $\mathrm{wt}(n', n) \in \mathbb{R}$ indicates connection strength from n' to n
- Incoming edges of n contribute to activation of n
  (All input potentials moderated by connection strength added up)



Input potential

$$1 \cdot \frac{3}{5} + \frac{\cancel{3}}{4} \cdot \frac{1}{\cancel{3}} + \frac{\cancel{3}}{\cancel{8}_4} \cdot \frac{\cancel{2}^1}{\cancel{3}} = \frac{3}{5} + \frac{1}{2} = 1.1$$

Neural networks

Formalization

- Given state $s\colon N \to \mathbb{R}$ and activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$
- Input potential supplied to σ yields new state $s'\colon N \to \mathbb{R}$

$$s' = \sigma\langle \mathrm{wt} \cdot s \rangle \qquad\qquad s'(n) = \sigma\Big\langle \sum_{(n',n)\in E} \mathrm{wt}(n',n) \cdot s(n') \Big\rangle$$

for all $n \in N$

Neural networks

Formalization

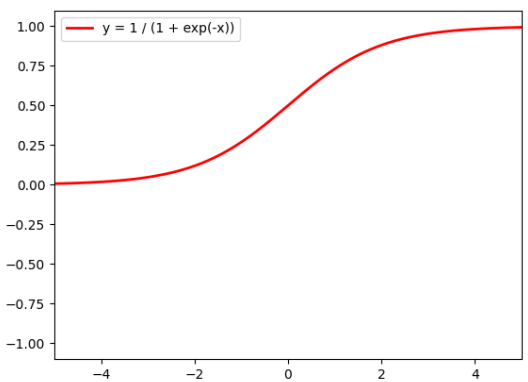
- Given state $s\colon N \to \mathbb{R}$ and activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$
- Input potential supplied to σ yields new state $s'\colon N \to \mathbb{R}$

$$s' = \sigma\langle \mathrm{wt} \cdot s \rangle \qquad s'(n) = \sigma\Big\langle \sum_{(n',n)\in E} \mathrm{wt}(n',n) \cdot s(n') \Big\rangle$$

for all $n \in N$

Standard activation functions

Sigmoid



Neural networks

Formalization

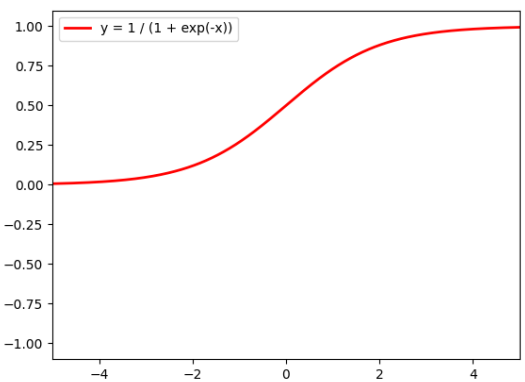
- Given state $s\colon N \to \mathbb{R}$ and activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$
- Input potential supplied to σ yields new state $s'\colon N \to \mathbb{R}$

$$s' = \sigma\langle \mathrm{wt} \cdot s \rangle \qquad s'(n) = \sigma\Big\langle \sum_{(n',n)\in E} \mathrm{wt}(n',n) \cdot s(n') \Big\rangle$$

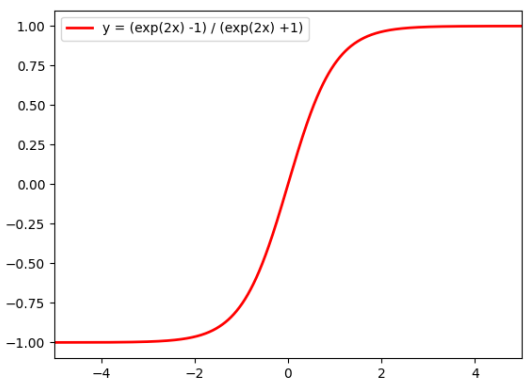
for all $n \in N$

Standard activation functions

Sigmoid



Tanh



Neural networks

Formalization

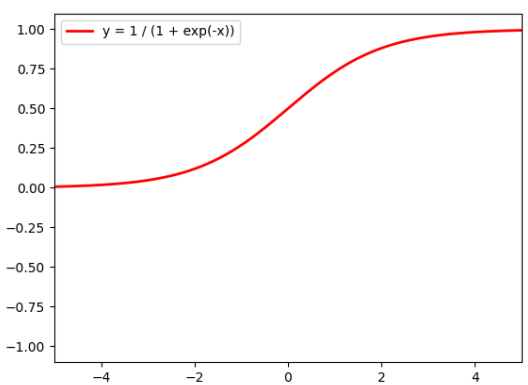
- Given state $s\colon N \to \mathbb{R}$ and activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$
- Input potential supplied to σ yields new state $s'\colon N \to \mathbb{R}$

$$s' = \sigma\langle \mathrm{wt} \cdot s \rangle \qquad s'(n) = \sigma\Big\langle \sum_{(n',n)\in E} \mathrm{wt}(n',n) \cdot s(n') \Big\rangle$$

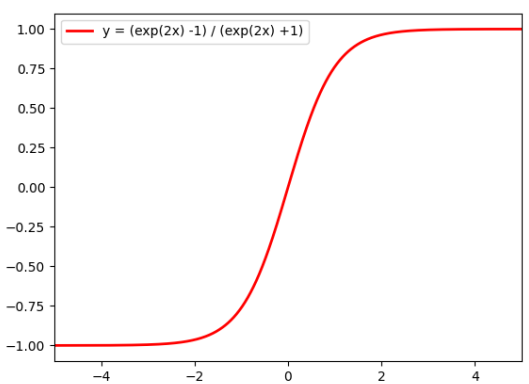
for all $n \in N$

Standard activation functions

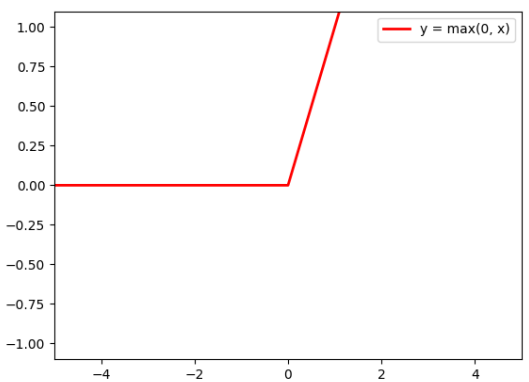
Sigmoid



Tanh

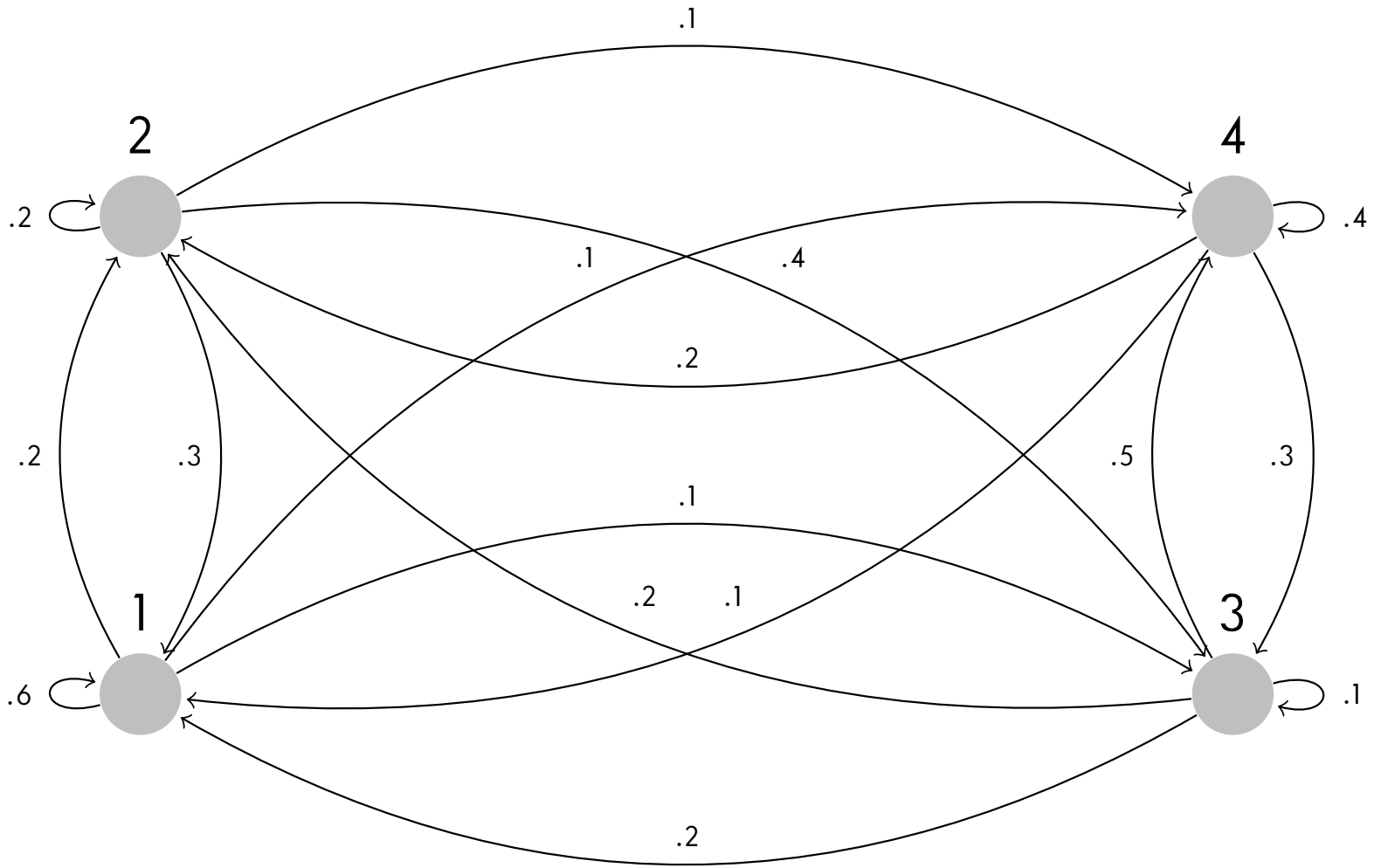


ReLU



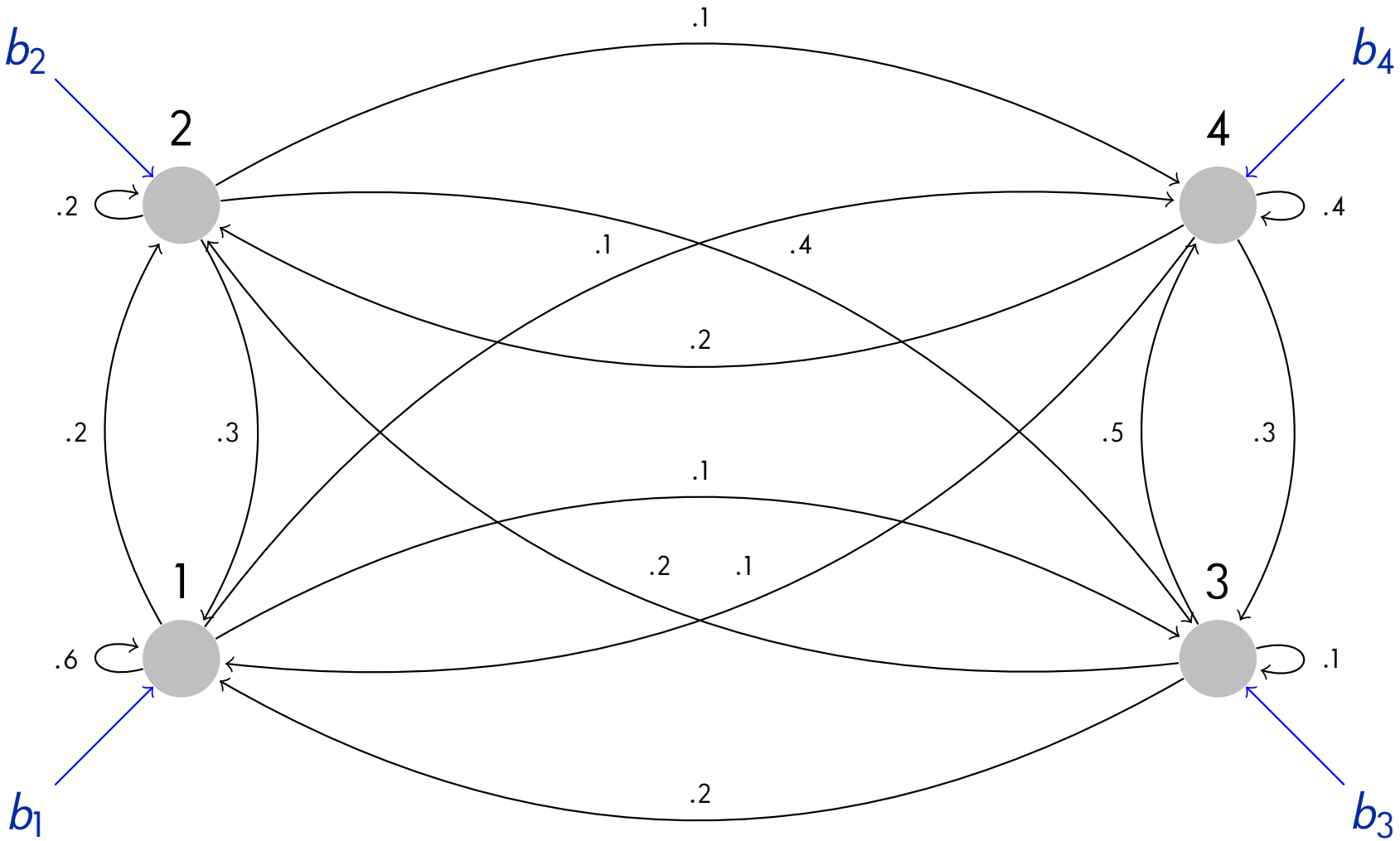
Neural networks

RNN = edge-labeled complete graph $(N, N \times N, \text{wt})$ with $\text{wt} \in \mathbb{R}^{N \times N}$



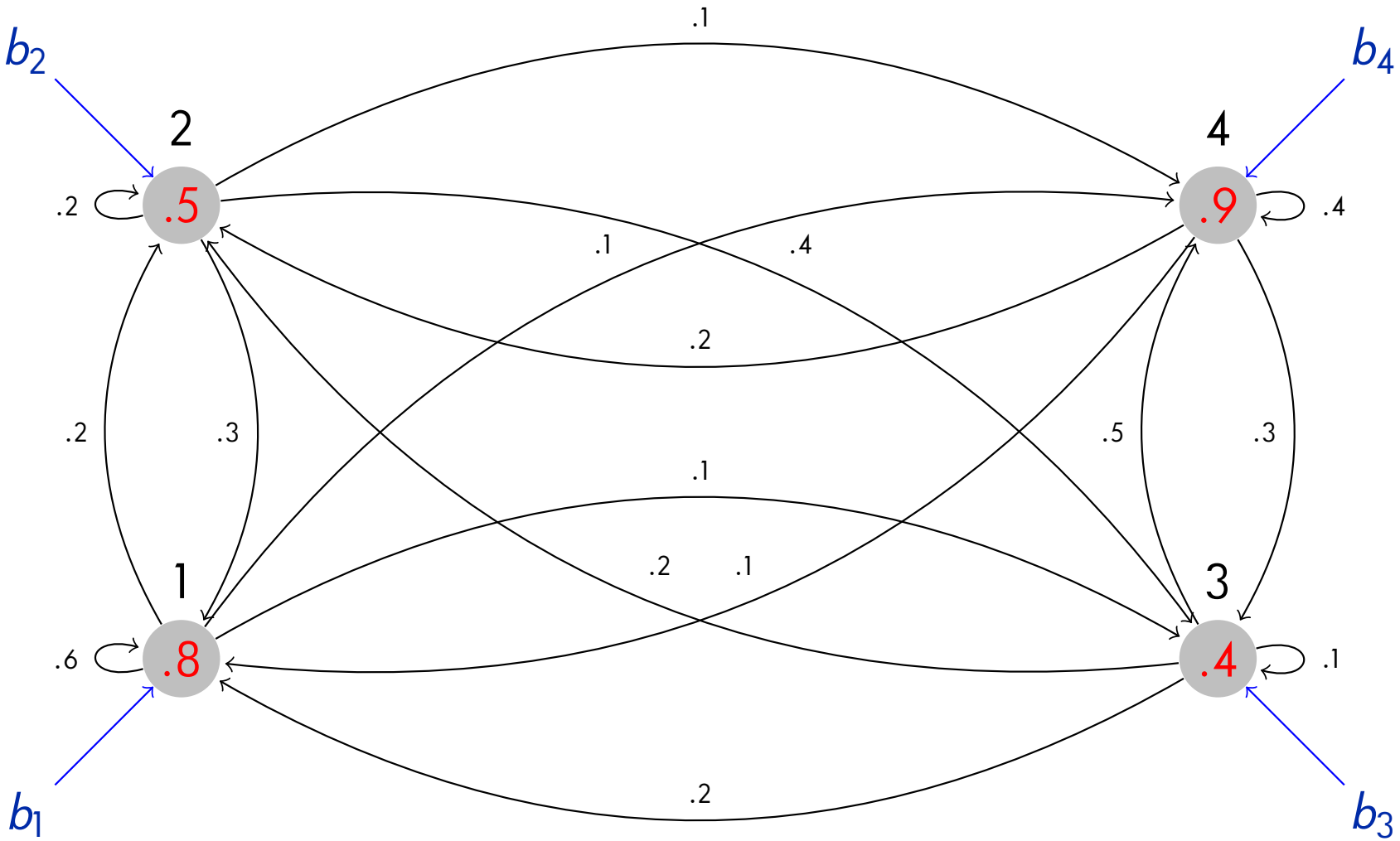
Neural networks

RNN = edge-labeled complete graph $(N, N \times N, \mathrm{wt})$ with $\mathrm{wt} \in \mathbb{R}^{N \times N}$



Neural networks

RNN = edge-labeled complete graph $(N, N \times N, \text{wt})$ with $\text{wt} \in \mathbb{R}^{N \times N}$



Neural networks

State behavior

- Edge weight matrix $\mathsf{wt} \in \mathbb{R}^{N \times N}$ and input vector $\mathsf{inp} \in \mathbb{R}^{N}$

$$\mathsf{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad \mathsf{inp} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Neural networks

State behavior

- Edge weight matrix $\mathsf{wt} \in \mathbb{R}^{N \times N}$ and input vector $\mathsf{inp} \in \mathbb{R}^{N}$

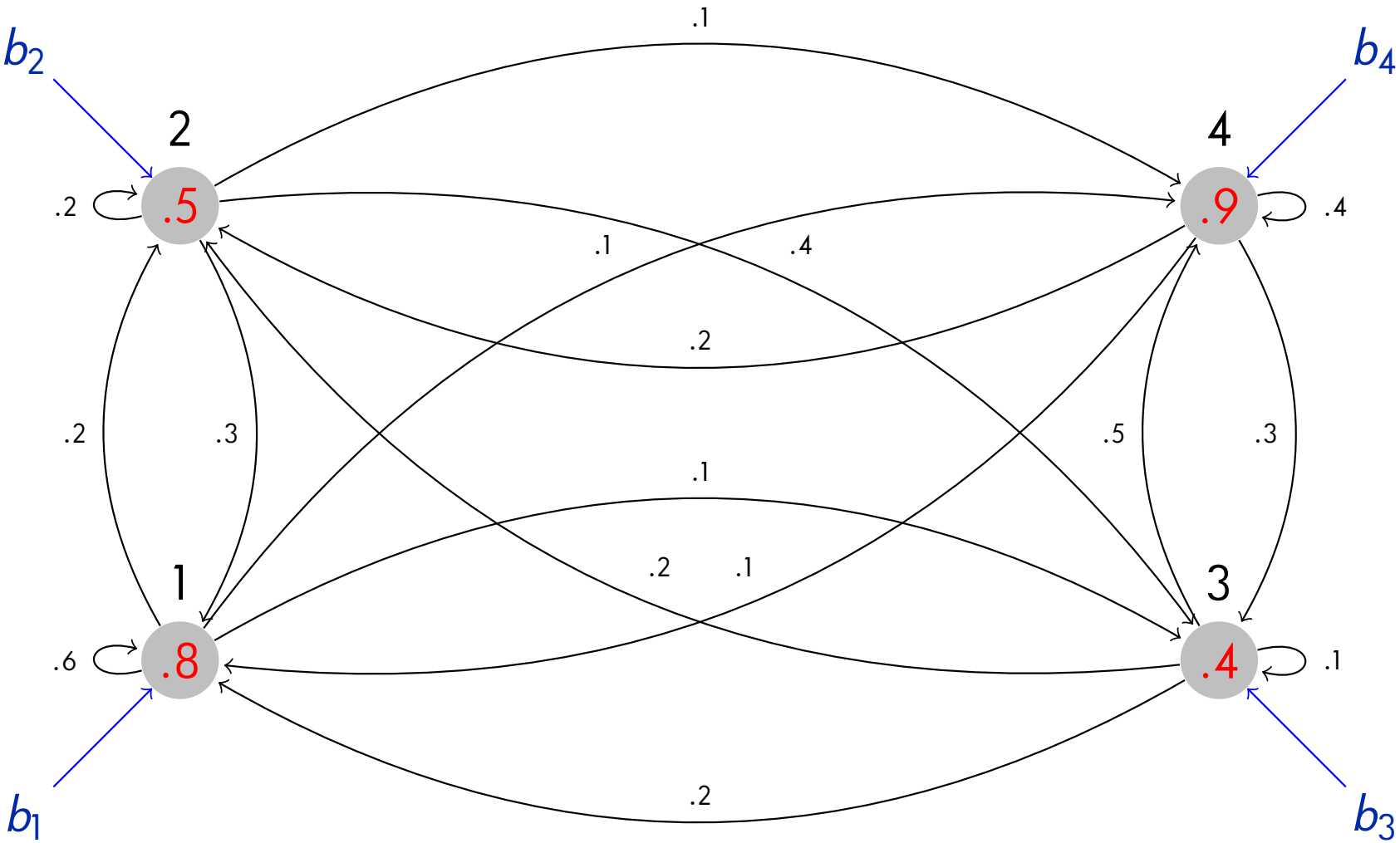
$$\mathsf{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad \mathsf{inp} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

- State vector $s \in \mathbb{R}^{N}$ and ReLU activation $\sigma(x) = \max(0, x)$
- Follow-state vector $s' \in \mathbb{R}^{N}$

$$s' = \sigma\langle \mathsf{inp} + \mathsf{wt} \cdot s \rangle \qquad s'(n) = \sigma\left\langle \mathsf{inp}(n) + \sum_{n' \in N} \mathsf{wt}(n', n) \cdot s(n') \right\rangle$$

for all $n \in N$

Neural networks



Neural networks

Example

$$\text{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad s = \begin{pmatrix} .25 \\ .25 \\ .25 \\ .25 \end{pmatrix} \qquad \text{inp} = \begin{pmatrix} -.8 \\ .5 \\ .4 \\ -.3 \end{pmatrix}$$

Neural networks

Example

$$\text{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad s = \begin{pmatrix} .25 \\ .25 \\ .25 \\ .25 \end{pmatrix} \qquad \text{inp} = \begin{pmatrix} -.8 \\ .5 \\ .4 \\ -.3 \end{pmatrix}$$

Then

$$\text{wt} \cdot s = \begin{pmatrix} .3 \\ .2 \\ .225 \\ .275 \end{pmatrix}$$

Neural networks

Example

$$\text{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad s = \begin{pmatrix} .25 \\ .25 \\ .25 \\ .25 \end{pmatrix} \qquad \text{inp} = \begin{pmatrix} -.8 \\ .5 \\ .4 \\ -.3 \end{pmatrix}$$

Then

$$\text{wt} \cdot s = \begin{pmatrix} .3 \\ .2 \\ .225 \\ .275 \end{pmatrix} \qquad \text{inp} + \text{wt} \cdot s = \begin{pmatrix} -.5 \\ .7 \\ .625 \\ -.025 \end{pmatrix}$$

Neural networks

Example

$$\mathrm{wt} = \begin{pmatrix} .6 & .3 & .2 & .1 \\ .2 & .2 & .2 & .2 \\ .1 & .4 & .1 & .3 \\ .1 & .1 & .5 & .4 \end{pmatrix} \qquad s = \begin{pmatrix} .25 \\ .25 \\ .25 \\ .25 \end{pmatrix} \qquad \mathrm{inp} = \begin{pmatrix} -.8 \\ .5 \\ .4 \\ -.3 \end{pmatrix}$$

Then

$$\mathrm{wt} \cdot s = \begin{pmatrix} .3 \\ .2 \\ .225 \\ .275 \end{pmatrix} \qquad \mathrm{inp} + \mathrm{wt} \cdot s = \begin{pmatrix} -.5 \\ .7 \\ .625 \\ -.025 \end{pmatrix}$$

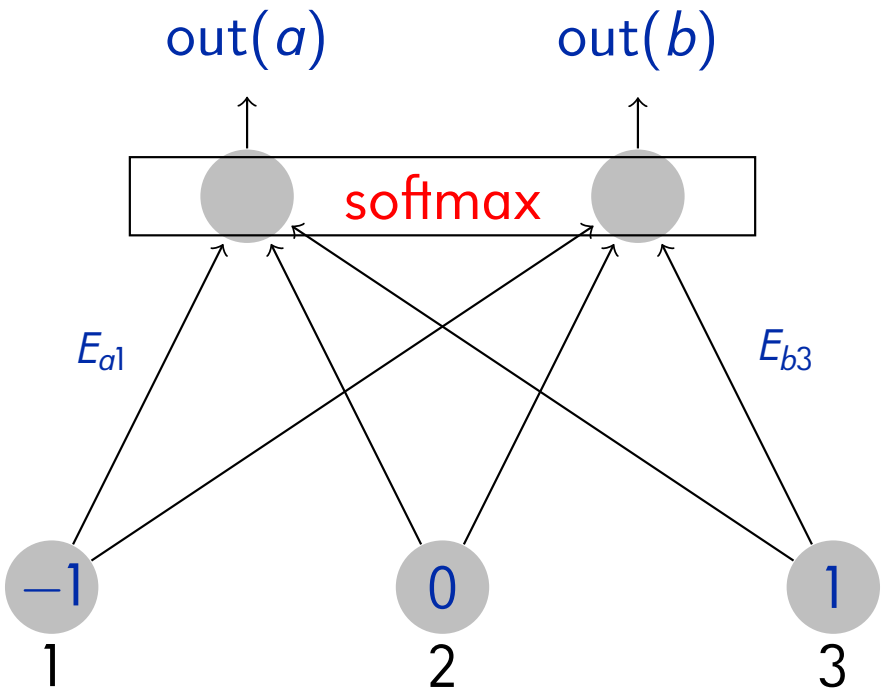
Applying ReLU activation σ

$$s' = \sigma\langle \mathrm{inp} + \mathrm{wt} \cdot s \rangle = \begin{pmatrix} 0 & .7 & .625 & 0 \end{pmatrix}^{\mathsf{T}}$$

Neural networks

Adding output

- Prediction matrix $E \in \mathbb{R}^{\Sigma \times N}$ to produce vector entry for each letter

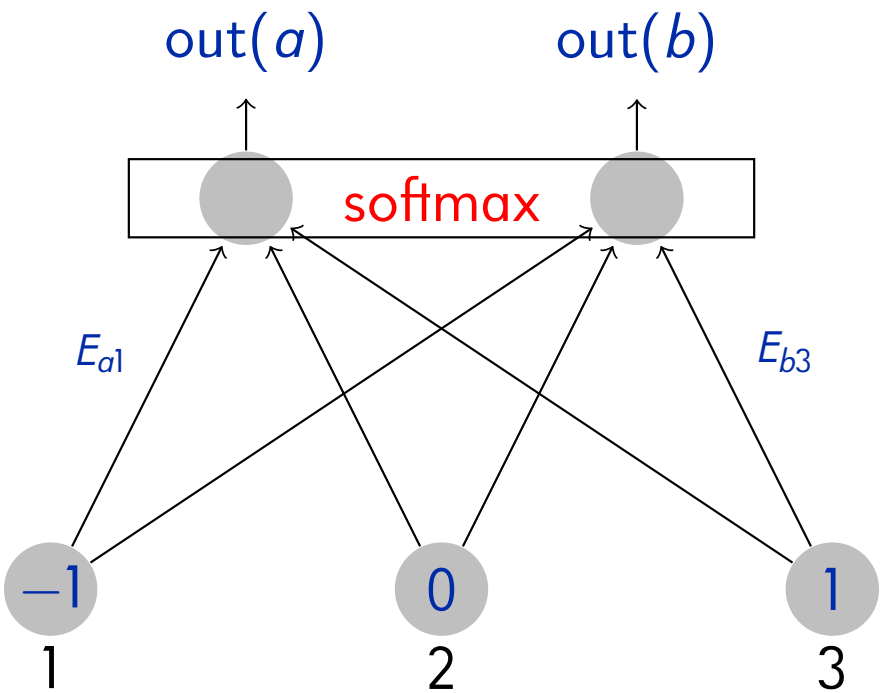


Neural networks

Adding output

- Prediction matrix $E \in \mathbb{R}^{\Sigma \times N}$ to produce vector entry for each letter
- Turn into probability via softmax: $\mathbb{R}^{\Sigma} \to \mathbb{R}^{\Sigma}$ for all $v \in \mathbb{R}^{\Sigma}$ and $a \in \Sigma$

$$\text{softmax}(v)_a = \frac{e^{v_a}}{\sum_{b \in \Sigma} e^{v_b}}$$



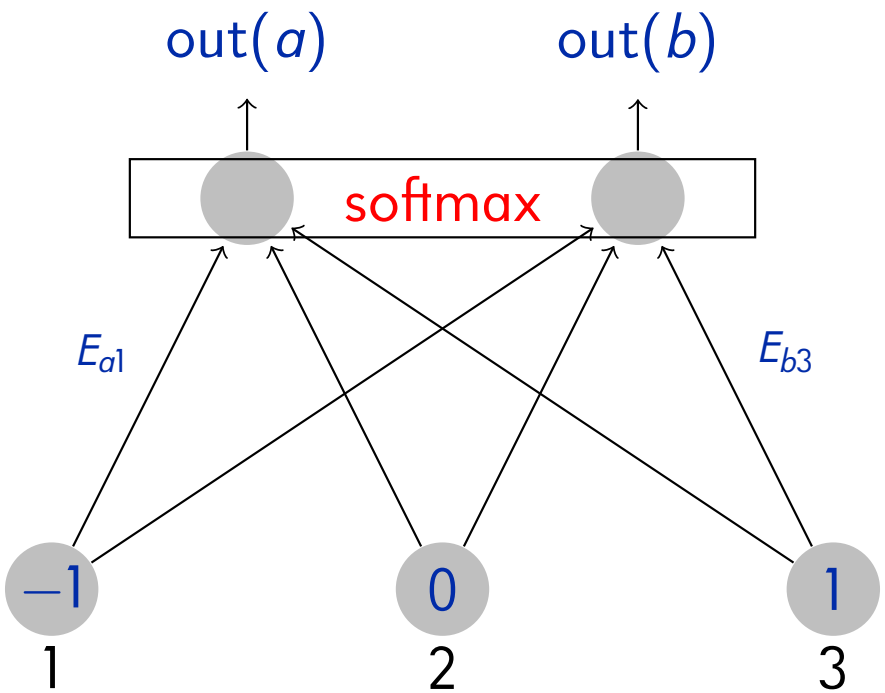
Neural networks

Adding output

- Prediction matrix $E \in \mathbb{R}^{\Sigma \times N}$ to produce vector entry for each letter
- Turn into probability via softmax: $\mathbb{R}^{\Sigma} \to \mathbb{R}^{\Sigma}$ for all $v \in \mathbb{R}^{\Sigma}$ and $a \in \Sigma$

$$\text{softmax}(v)_a = \frac{e^{v_a}}{\sum_{b \in \Sigma} e^{v_b}}$$

- Letter classification $\text{out} = \text{softmax}\langle E \cdot s \rangle$

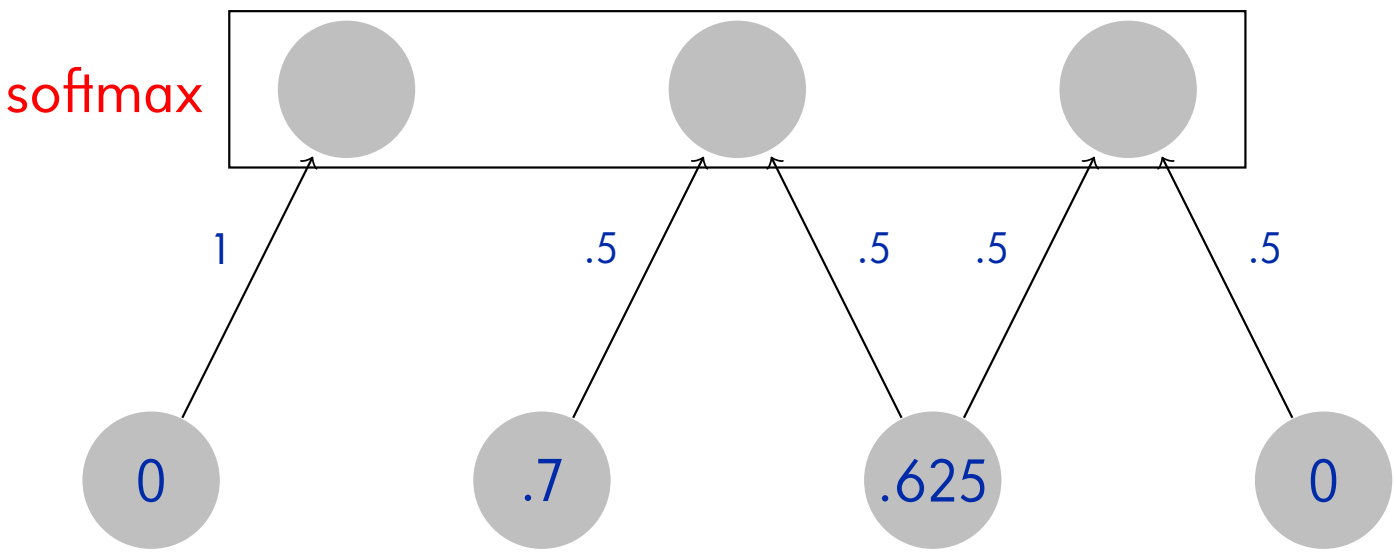


Neural networks

Example

- State $s = \begin{pmatrix} 0 & .7 & .625 & 0 \end{pmatrix}^\mathsf{T}$ and prediction $E = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & .5 & .5 & 0 \\ 0 & 0 & .5 & .5 \end{pmatrix}$

$$\text{softmax}\langle E \cdot s \rangle = \text{softmax}\left\langle \begin{matrix} 0 \\ .6625 \\ .3125 \end{matrix} \right\rangle \approx \begin{pmatrix} .23 \\ .45 \\ .32 \end{pmatrix}$$



Neural networks

Definition (RNN)

(Single-layer) RNN is $\left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_a)_{a \in \Sigma_\$}, E\right)$

- finite set N and alphabet Σ
- initial state $s_{-1} \in \mathbb{Q}^N$ and transition weights $\mathrm{wt} \in \mathbb{Q}^{N \times N}$
- input vector $\mathrm{inp}_\sigma \in \mathbb{Q}^N$ for every $\sigma \in \Sigma_\$$ with $\Sigma_\$ = \Sigma \cup \{\$\}$
- prediction matrix $E \in \mathbb{Q}^{\Sigma_\$ \times N}$

Neural networks

Definition (Semantics of RNN)

Let $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\sigma)_{\sigma \in \Sigma_\$}, E\right)$ be RNN and $w = a_1 \cdots a_k \in \Sigma^*$

- $a_0 = \$$, $a_{k+1} = \$$, and $s_{-1}^w = s_{-1}$
- State $s_t^w = \sigma\langle \mathrm{inp}_{a_t} + \mathrm{wt} \cdot s_{t-1}^w \rangle$ for every $0 \leq t \leq k$

Neural networks

Definition (Semantics of RNN)

Let $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\sigma)_{\sigma \in \Sigma_\$}, E\right)$ be RNN and $w = a_1 \cdots a_k \in \Sigma^*$

- $a_0 = \$$, $a_{k+1} = \$$, and $s^w_{-1} = s_{-1}$
- State $s^w_t = \sigma\langle \mathrm{inp}_{a_t} + \mathrm{wt} \cdot s^w_{t-1} \rangle$ for every $0 \leq t \leq k$
- Output $\mathrm{out}^w_t = \mathrm{softmax}\langle E \cdot s^w_t \rangle$ for every $0 \leq t \leq k$

Neural networks

Definition (Semantics of RNN)

Let $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\sigma)_{\sigma \in \Sigma_\$}, E\right)$ be RNN and $w = a_1 \cdots a_k \in \Sigma^*$

- $a_0 = \$$, $a_{k+1} = \$$, and $s_{-1}^w = s_{-1}$
- State $s_t^w = \sigma\langle \mathrm{inp}_{a_t} + \mathrm{wt} \cdot s_{t-1}^w \rangle$ for every $0 \leq t \leq k$
- Output $\mathrm{out}_t^w = \mathrm{softmax}\langle E \cdot s_t^w \rangle$ for every $0 \leq t \leq k$
- Weight of w

$$\mathcal{R}(w) = \prod_{t=0}^{k} \mathrm{out}_t^w(a_{t+1})$$

Neural networks

Definition (Semantics of RNN)

Let $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\sigma)_{\sigma \in \Sigma_\$}, E\right)$ be RNN and $w = a_1 \cdots a_k \in \Sigma^*$

- $a_0 = \$$, $a_{k+1} = \$$, and $s_{-1}^w = s_{-1}$
- State $s_t^w = \sigma\langle \mathrm{inp}_{a_t} + \mathrm{wt} \cdot s_{t-1}^w \rangle$ for every $0 \leq t \leq k$
- Output $\mathrm{out}_t^w = \mathrm{softmax}\langle E \cdot s_t^w \rangle$ for every $0 \leq t \leq k$
- Weight of w

$$\mathcal{R}(w) = \prod_{t=0}^{k} \mathrm{out}_t^w(a_{t+1})$$

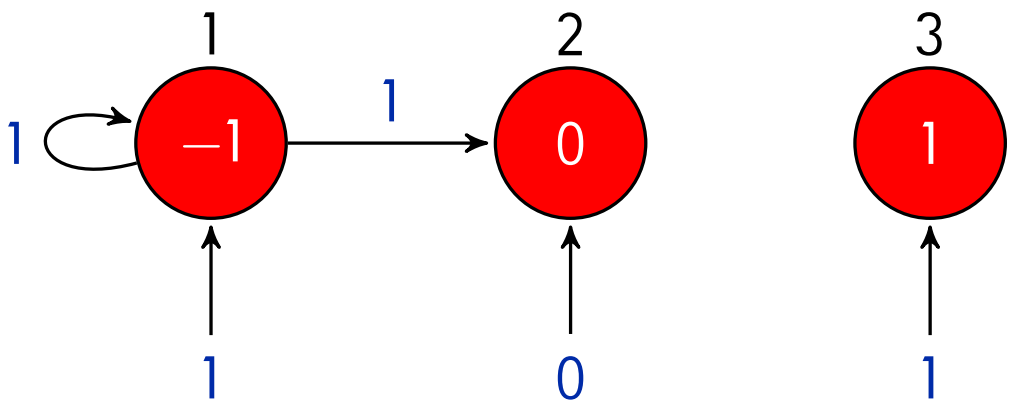
Note

- Thus $\mathcal{R} \colon \Sigma^* \to [0,1]$ with $\sum_{w \in \Sigma^*} \mathcal{R}(w) \leq 1$

Neural networks

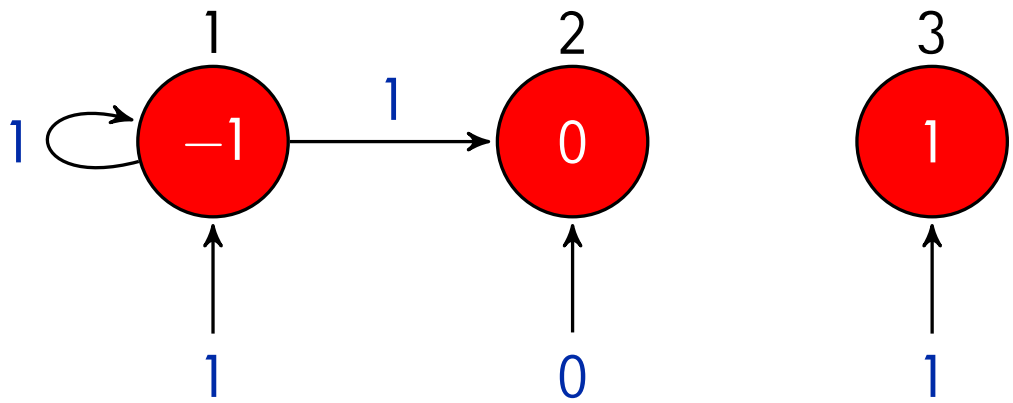
Example RNN $\mathcal{R} = \big(\{1,2,3\}, \{a\}, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\$, \mathrm{inp}_a), E\big)$ and $z \in \mathbb{Q}$

$$\mathrm{wt} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad s_{-1} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \qquad \mathrm{inp}_\$ = \mathrm{inp}_a = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$



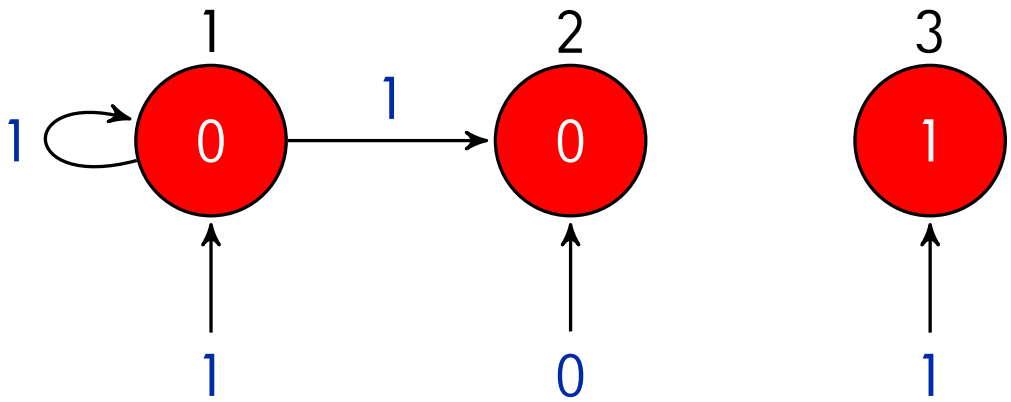
Neural networks

Example (cont'd)



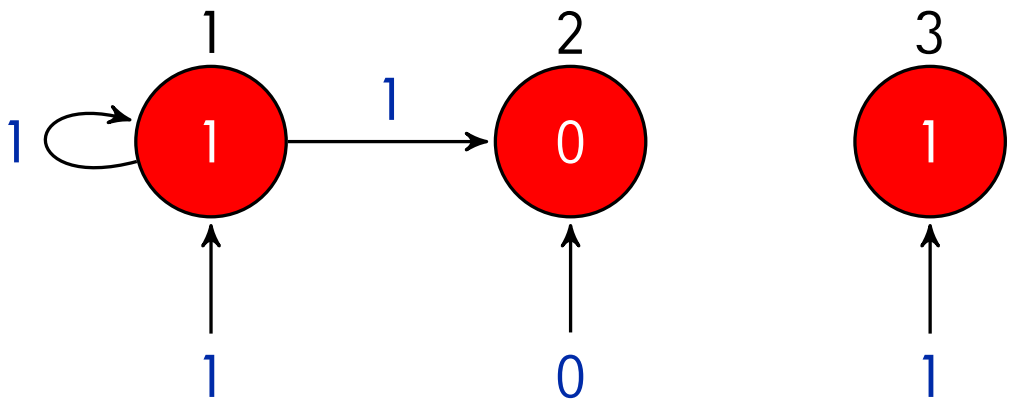
Neural networks

Example (cont'd)



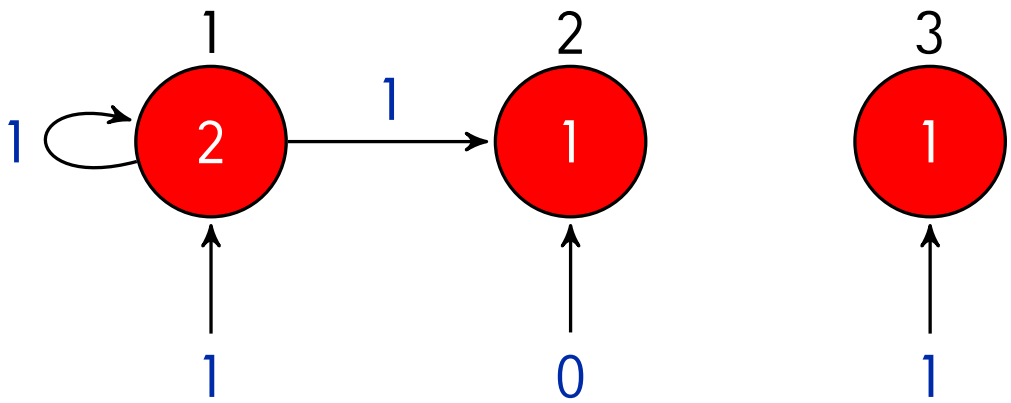
Neural networks

Example (cont'd)



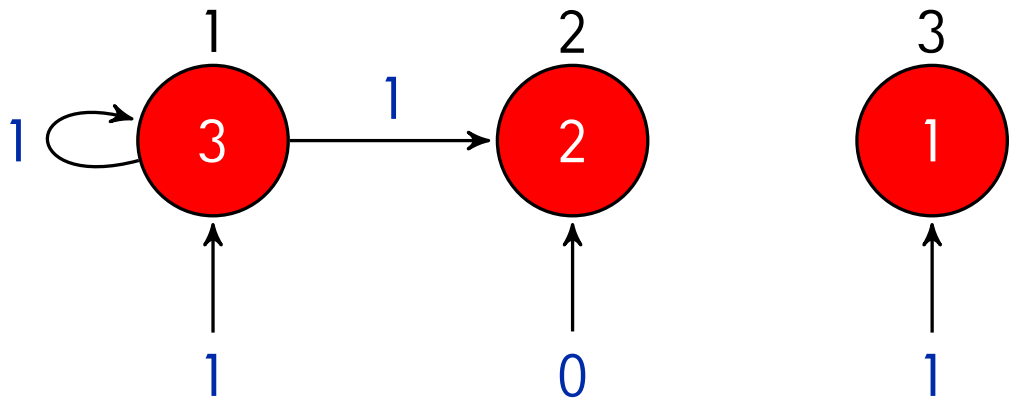
Neural networks

Example (cont'd)



Neural networks

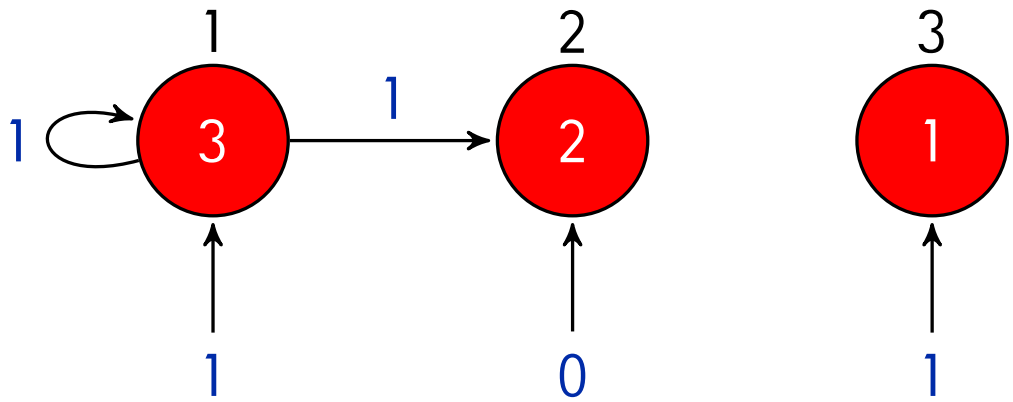
Example (cont'd)



- Input $w = a^k$

Neural networks

Example (cont'd)

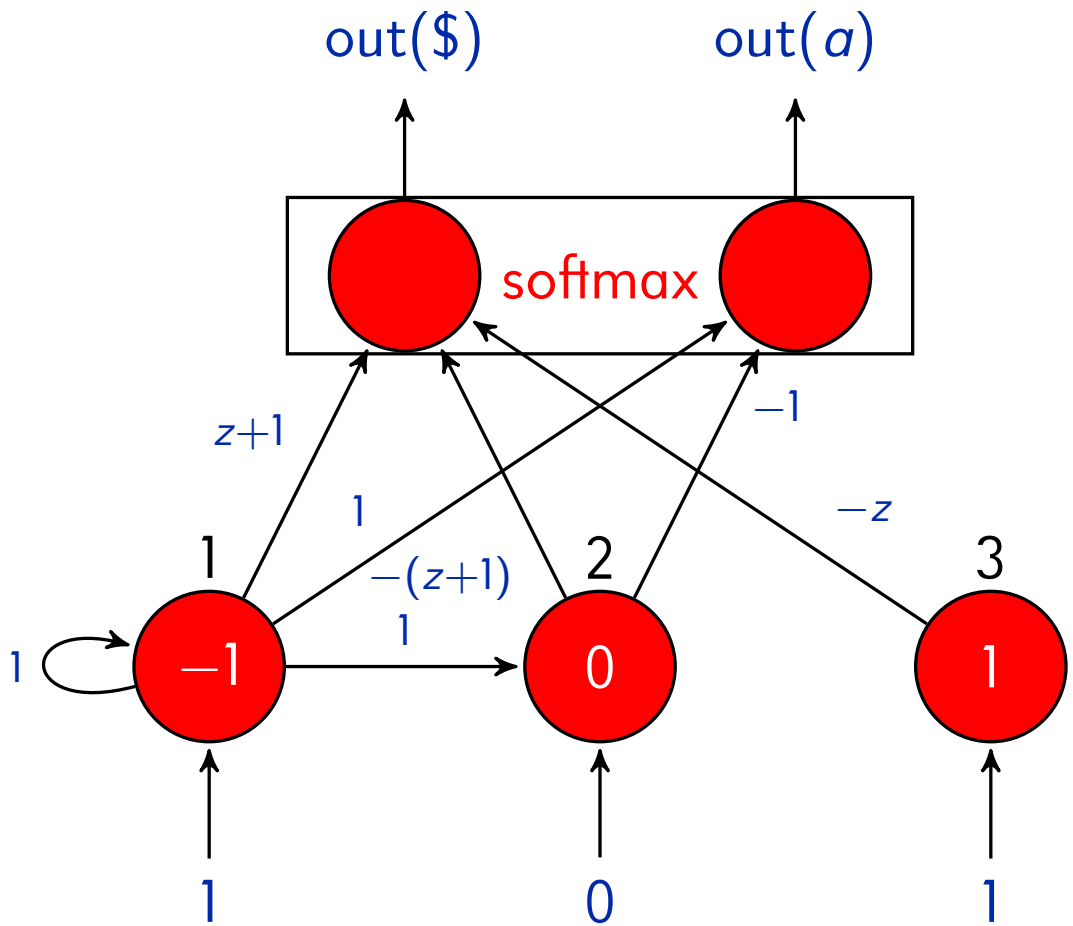


- Input $w = a^k$
- States $s_t^w = \sigma\begin{pmatrix} 1 + s_{t-1}^w(1) \\ s_{t-1}^w(1) \\ 1 \end{pmatrix} = \sigma\begin{pmatrix} t \\ t-1 \\ 1 \end{pmatrix}$ for all $0 \leq t \leq k$

Neural networks

Example (cont'd)

- Prediction matrix $E = \begin{pmatrix} z+1 & -(z+1) & -z \\ 1 & -1 & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$



Neural networks

Example (cont'd)

- States $s_t^w = \sigma\left\langle \begin{matrix} t \\ t-1 \\ 1 \end{matrix} \right\rangle$ for all $0 \leq t \leq k$

Neural networks

Example (cont'd)

- States $s_t^w = \sigma\left\langle \begin{matrix} t \\ t-1 \\ 1 \end{matrix} \right\rangle$ for all $0 \leq t \leq k$

- Prediction matrix $E = \begin{pmatrix} z+1 & -(z+1) & -z \\ 1 & -1 & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

Neural networks

Example (cont'd)

- States $s_t^w = \sigma\left\langle \begin{matrix} t \\ t-1 \\ 1 \end{matrix} \right\rangle$ for all $0 \le t \le k$
- Prediction matrix $E = \begin{pmatrix} z+1 & -(z+1) & -z \\ 1 & -1 & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$
- $\text{out}_t^w = \text{softmax}\left\langle \begin{matrix} (z+1)t - (z+1)\sigma\langle t-1\rangle - z \\ t - \sigma\langle t-1\rangle \end{matrix} \right\rangle$ for all $0 \le t \le k$

Neural networks

Example (cont'd)

- States $s_t^w = \sigma\left\langle \begin{matrix} t \\ t-1 \\ 1 \end{matrix} \right\rangle$ for all $0 \le t \le k$
- Prediction matrix $E = \begin{pmatrix} z+1 & -(z+1) & -z \\ 1 & -1 & 0 \end{pmatrix}$ $\$ \; a$
- $\text{out}_t^w = \text{softmax}\left\langle \begin{matrix} (z+1)t - (z+1)\sigma\langle t-1\rangle - z \\ t - \sigma\langle t-1\rangle \end{matrix} \right\rangle$ for all $0 \le t \le k$
- $\text{out}_0^w = \text{softmax}\left\langle \begin{matrix} -z \\ 0 \end{matrix} \right\rangle$ and $\text{out}_t^w = \text{softmax}\left\langle \begin{matrix} 1 \\ 1 \end{matrix} \right\rangle$ for all $1 \le t \le k$

Example (cont'd)

- $\text{out}_0^w = \text{softmax}\left\langle \begin{matrix} -z \\ 0 \end{matrix} \right\rangle$ and $\text{out}_t^w = \text{softmax}\left\langle \begin{matrix} 1 \\ 1 \end{matrix} \right\rangle$ for all $1 \leq t \leq k$

Neural networks

Example (cont'd)

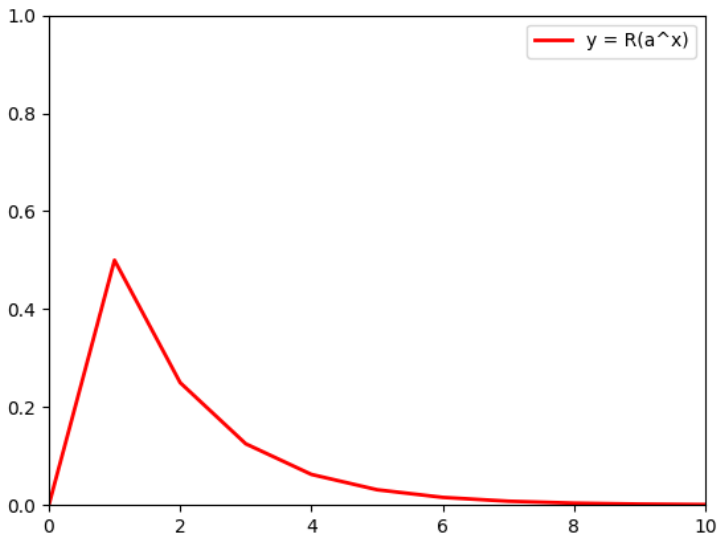
- $\text{out}_0^w = \text{softmax}\left\langle \begin{matrix} -z \\ 0 \end{matrix} \right\rangle$ and $\text{out}_t^w = \text{softmax}\left\langle \begin{matrix} 1 \\ 1 \end{matrix} \right\rangle$ for all $1 \leq t \leq k$

- Computed weights

$$\mathcal{R}(\varepsilon) = \frac{e^{-z}}{1 + e^{-z}} = \frac{1}{1 + e^{z}}$$

$$\mathcal{R}(a^k) = \frac{1}{1 + e^{-z}} \cdot 2^{-k}$$

for $k \neq 0$



(graph for $z = 100$)

Neural networks

Example RNN $\mathcal{R} = \big(\{1,2,3,4\}, \{a\}, s_{-1}, \mathrm{wt}, (\mathrm{inp}_\$, \mathrm{inp}_a), E\big)$ and $z \in \mathbb{Q}$

$$\mathrm{wt} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad s_{-1} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad \mathrm{inp}_\$ = \mathrm{inp}_a = \begin{pmatrix} -99 \\ -100 \\ 1 \\ 1 \end{pmatrix}$$

States for input $w = a^k$

$$s_t^w = \sigma \begin{pmatrix} t-99 \\ t-100 \\ t+1 \\ 1 \end{pmatrix} \qquad \text{for all } 0 \leq t \leq k$$

Example (cont'd)

- States

$$s_t^w = \sigma \begin{pmatrix} t-99 \\ t-100 \\ t+1 \\ 1 \end{pmatrix} \qquad \text{for all } 0 \leq t \leq k$$

- Prediction matrix $E = \begin{pmatrix} z & -z & 0 & -z \\ -z & z & 0 & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

Neural networks

Example (cont'd)

- States

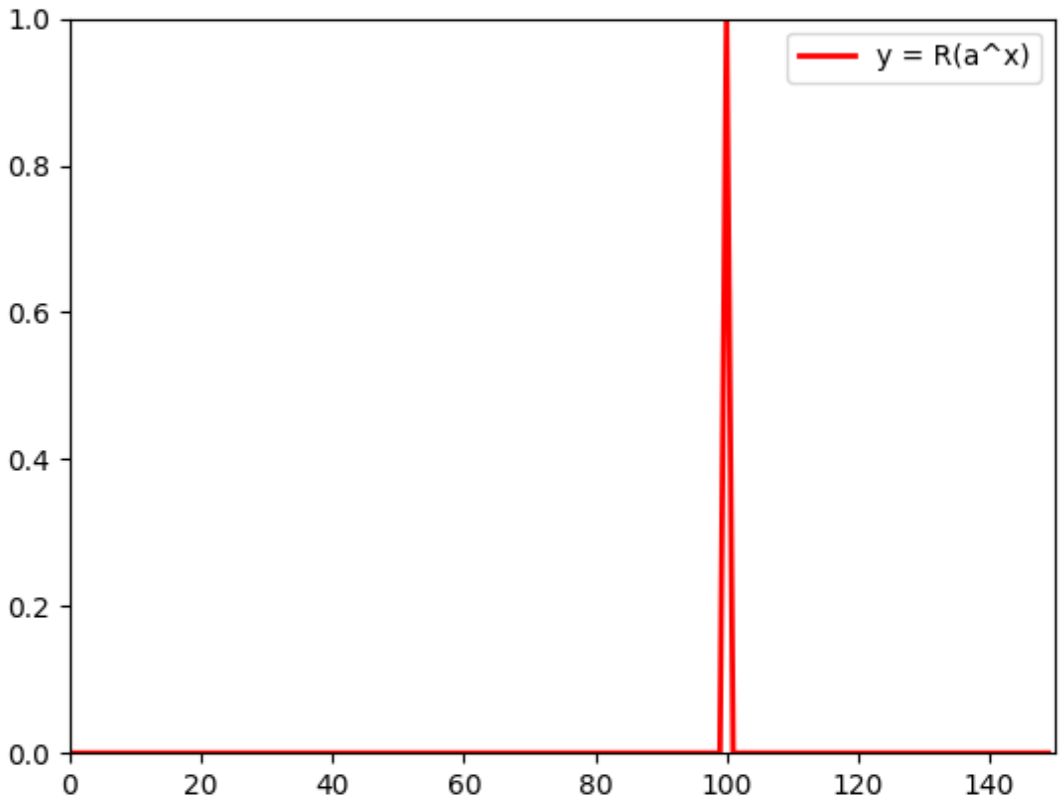
$$s_t^w = \sigma\begin{pmatrix} t-99 \\ t-100 \\ t+1 \\ 1 \end{pmatrix} \qquad \text{for all } 0 \le t \le k$$

- Prediction matrix $E = \begin{pmatrix} z & -z & 0 & -z \\ -z & z & 0 & 0 \end{pmatrix}$ $\$$ a

- $\text{out}_t^w = \text{softmax}\begin{pmatrix} -z \\ 0 \end{pmatrix}$ for $0 \le t \le 99$

  $\text{out}_t^w = \text{softmax}\begin{pmatrix} 0 \\ -z \end{pmatrix}$ for all $100 \le t \le k$

Example (cont'd)



(graph for $z = 100$)

Expressive power

Theorem [Siegelmann & Sontag 1995]

For every deterministic Turing machine M there exists RNN

$$(N, \{a\}, s_{-1}, \mathrm{wt}, (\mathrm{inp}_{\$}, \mathrm{inp}_a), E)$$

and 2 designated neurons $n, n' \in N$ such that for all $w = a^k$ and $0 \leq t \leq k$

- $s_t^w(n) - s_t^w(n') \in \{0, 1\}$
- $s_t^w(n) - s_t^w(n') = 1$ iff M halts on ε after exactly t steps

Decidability

Definition (Best string)

Best string for RNN $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_a)_{a \in \Sigma_\$}, E\right)$ is

$$\widehat{w} = \arg\max_{w \in \Sigma^*} \mathcal{R}(w)$$

Best string problem: Is $\mathcal{R}(\widehat{w}) \geq \theta$ given $\mathcal{R}$ and $\theta \in [0,1] \cap \mathbb{Q}$

Decidability

Definition (Best string)

Best string for RNN $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_a)_{a \in \Sigma_\$}, E\right)$ is

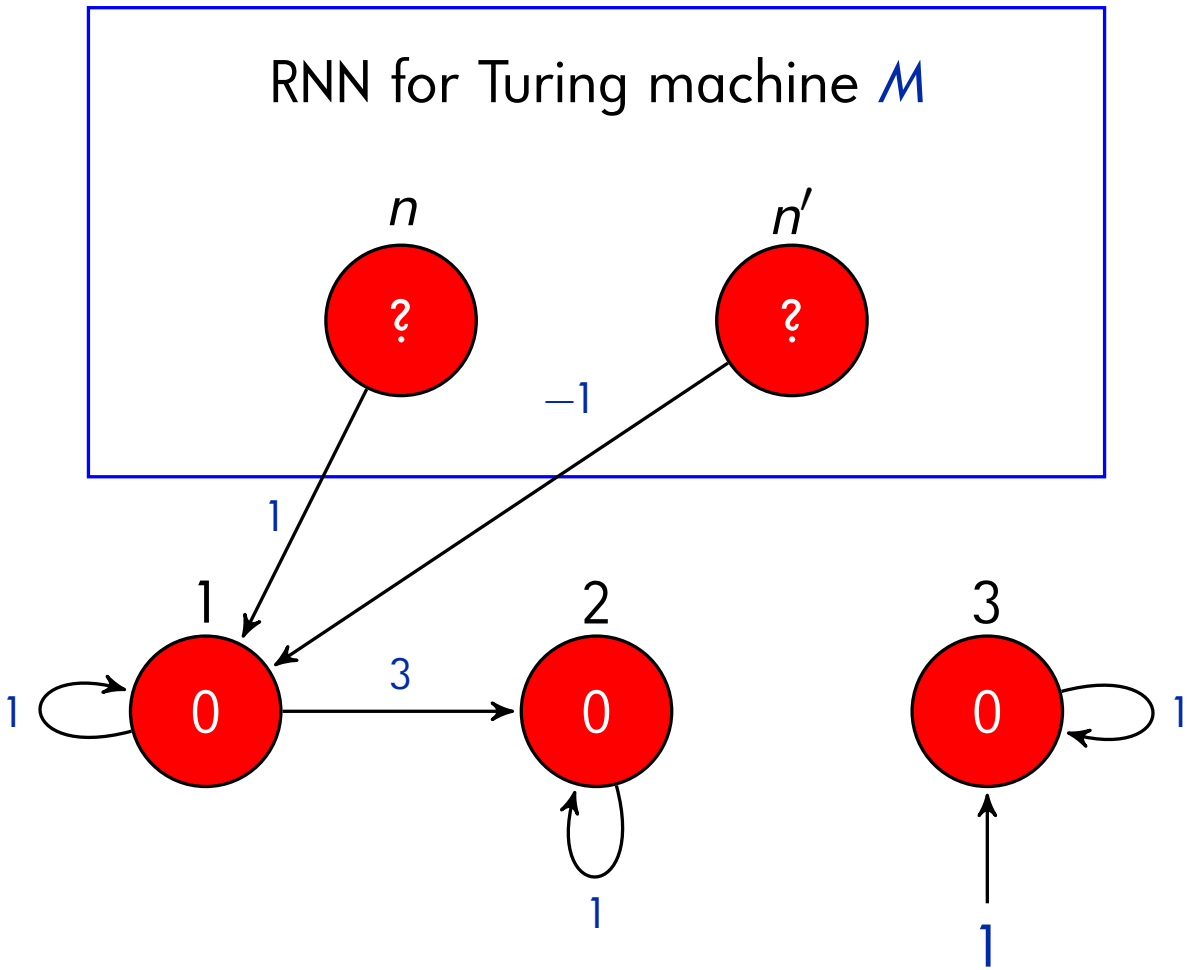
$$\widehat{w} = \arg\max_{w \in \Sigma^*} \mathcal{R}(w)$$

Best string problem: Is $\mathcal{R}(\widehat{w}) \geq \theta$ given $\mathcal{R}$ and $\theta \in [0,1] \cap \mathbb{Q}$

Theorem [CGMMK 2018]

Best string problem undecidable for RNN

Decidability



Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 1: TM M does not halt

$$s_t^w = \begin{pmatrix} 0 \\ 0 \\ t+1 \\ \vdots \end{pmatrix}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 1: TM M does not halt

$$s_t^w = \begin{pmatrix} 0 \\ 0 \\ t+1 \\ \vdots \end{pmatrix}$$

$$\text{out}_t^w(\$) = \frac{e^{-(t+1)}}{e^{-(t+1)} + e^{t+1}} = \frac{1}{1 + e^{2(t+1)}} \leq \frac{1}{1 + e^2} < 0.12$$

Thus $\mathcal{R}(a^k) < 0.12$ for all $k \in \mathbb{N}$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 2: TM M halts after exactly ℓ steps

$$s_t^w = \begin{pmatrix} t \overset{?}{>} \ell \\ \sigma\langle 3(t-\ell-1)\rangle \\ t+1 \\ \vdots \end{pmatrix}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 2: TM M halts after exactly ℓ steps

$$s_t^w = \begin{pmatrix} t \overset{?}{>} \ell \\ \sigma\langle 3(t-\ell-1)\rangle \\ t+1 \\ \vdots \end{pmatrix}$$

$$\text{out}_t^w(\$) = \begin{cases} \frac{1}{1+e^{2(t+1)}} & \text{if } t \leq \ell \\ \frac{e^{t-3\ell-5}}{1+e^{t-3\ell-5}} & \text{otherwise} \end{cases}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 2: TM M halts after exactly ℓ steps

$$s_t^w = \begin{pmatrix} t \overset{?}{>} \ell \\ \sigma\langle 3(t-\ell-1)\rangle \\ t+1 \\ \vdots \end{pmatrix}$$

$$\mathrm{out}_t^w(\$) = \begin{cases} \frac{1}{1+e^{2(t+1)}} & \text{if } t \leq \ell \\ \frac{e^{t-3\ell-5}}{1+e^{t-3\ell-5}} & \text{otherwise} \end{cases}$$

$$\mathcal{R}(a^{3\ell-5}) = \left(\prod_{t=0}^{\ell} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\right) \cdot \left(\prod_{t=\ell+1}^{3\ell-6} \frac{1}{1+e^{t-3\ell-5}}\right) \cdot \frac{1}{2}$$

Definition (Infinite Pochhammer symbol)

Let $x \in \mathbb{R}$ and $q \in [0, 1]$. Infinite q-Pochhammer symbol is

$$(x; q)_\infty = \prod_{i \in \mathbb{N}} (1 - xq^i)$$

Decidability

Definition (Infinite Pochhammer symbol)

Let $x \in \mathbb{R}$ and $q \in [0,1]$. Infinite q-Pochhammer symbol is

$$(x;q)_\infty = \prod_{i \in \mathbb{N}} (1 - xq^i)$$

Examples

$$(-1;e^{-2})_\infty = \prod_{i \in \mathbb{N}} (1 + e^{-2i}) \approx 2.32$$

$$(-1;e^{-1})_\infty = \prod_{i \in \mathbb{N}} (1 + e^{-i}) \approx 3.36$$

Decidability

$$\mathcal{R}(a^{3\ell-5}) = \Big(\prod_{t=0}^{\ell} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{3\ell-6} \frac{1}{1+e^{t-3\ell-5}}\Big) \cdot \frac{1}{2}$$

Decidability

$$\mathcal{R}(a^{3\ell-5}) = \Big(\prod_{t=0}^{\ell} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{3\ell-6} \frac{1}{1+e^{t-3\ell-5}}\Big) \cdot \frac{1}{2}$$

$$= \Big(\frac{2}{2} \cdot \prod_{t=1}^{\ell+1} \frac{1}{1+e^{-2t}}\Big) \cdot \Big(\prod_{t=11}^{2\ell+4} \frac{1}{1+e^{-t}}\Big) \cdot \frac{1}{2}$$

Decidability

$$\begin{aligned}
\mathcal{R}(a^{3\ell-5}) &= \Big(\prod_{t=0}^{\ell} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{3\ell-6} \frac{1}{1+e^{t-3\ell-5}}\Big) \cdot \frac{1}{2} \\
&= \Big(\frac{\cancel{2}}{\cancel{2}} \cdot \prod_{t=1}^{\ell+1} \frac{1}{1+e^{-2t}}\Big) \cdot \Big(\prod_{t=11}^{2\ell+4} \frac{1}{1+e^{-t}}\Big) \cdot \frac{1}{\cancel{2}} \\
&\geq \Big(\prod_{t=0}^{\ell+1} \frac{1}{1+e^{-2t}}\Big) \cdot 2 \cdot \Big(\prod_{t=0}^{2\ell+4} \frac{1}{1+e^{-t}}\Big)
\end{aligned}$$

Decidability

$$\begin{aligned}
\mathcal{R}(a^{3\ell-5}) &= \Big(\prod_{t=0}^{\ell} \frac{e^{2(t+1)}}{1+e^{2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{3\ell-6} \frac{1}{1+e^{t-3\ell-5}}\Big) \cdot \frac{1}{2} \\
&= \Big(\frac{\cancel{2}}{2} \cdot \prod_{t=1}^{\ell+1} \frac{1}{1+e^{-2t}}\Big) \cdot \Big(\prod_{t=11}^{2\ell+4} \frac{1}{1+e^{-t}}\Big) \cdot \frac{1}{\cancel{2}} \\
&\geq \Big(\prod_{t=0}^{\ell+1} \frac{1}{1+e^{-2t}}\Big) \cdot 2 \cdot \Big(\prod_{t=0}^{2\ell+4} \frac{1}{1+e^{-t}}\Big) \\
&\geq \frac{2}{(-1;e^{-2})_\infty \cdot (-1;e^{-1})_\infty} \geq 0.25
\end{aligned}$$

So best tree has weight $\widehat{w} \geq 0.2$ iff M halts

Decidability

Definition (Consistency)

RNN $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_a)_{a \in \Sigma_\$}, E\right)$ is consistent if $\sum_{w \in \Sigma^*} \mathcal{R}(w) = 1$

Decidability

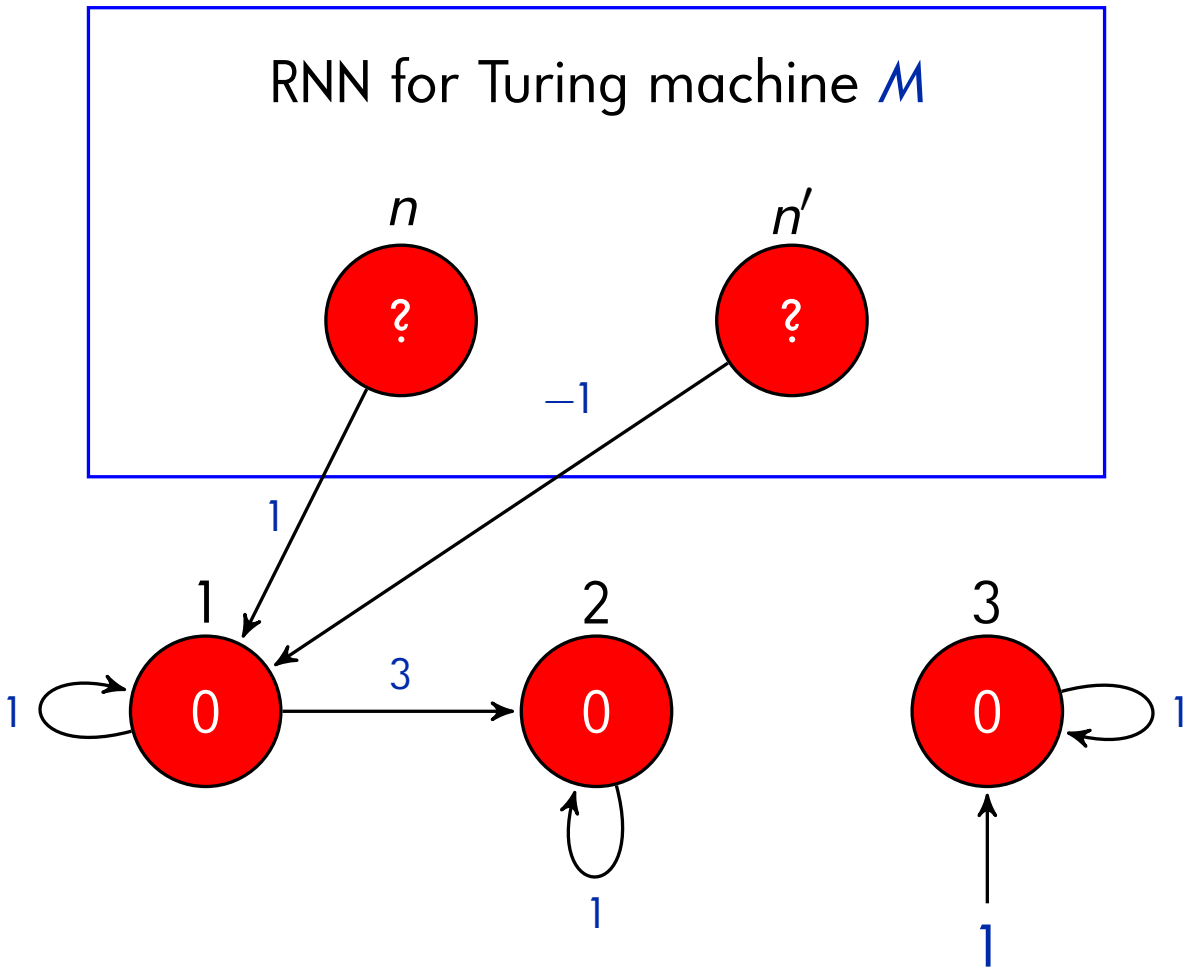
Definition (Consistency)

RNN $\mathcal{R} = \left(N, \Sigma, s_{-1}, \mathrm{wt}, (\mathrm{inp}_a)_{a \in \Sigma_\$}, E\right)$ is consistent if $\sum_{w \in \Sigma^*} \mathcal{R}(w) = 1$

Theorem [CGMMK 2018]

Consistency undecidable for RNN

Decidability



Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 1: TM M does not halt

$$s_t^w = \begin{pmatrix} 0 & 0 & t+1 & \cdots \end{pmatrix}^T \qquad \text{out}_t^w(\$) = \frac{1}{1+e^{2(t+1)}}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 1: TM M does not halt

$$s_t^w = \begin{pmatrix} 0 & 0 & t+1 & \cdots \end{pmatrix}^{\mathsf{T}} \qquad \text{out}_t^w(\$) = \frac{1}{1+\mathrm{e}^{2(t+1)}}$$

Thus $\mathcal{R}$ is inconsistent because

$$\mathcal{R}(a^k) = \frac{1}{1+\mathrm{e}^{2(k+1)}} \cdot \prod_{t=0}^{k-1} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 1: TM M does not halt

$$s_t^w = \begin{pmatrix} 0 & 0 & t+1 & \cdots \end{pmatrix}^{\mathsf{T}} \qquad \mathrm{out}_t^w(\$) = \frac{1}{1+\mathrm{e}^{2(t+1)}}$$

Thus $\mathcal{R}$ is inconsistent because

$$\mathcal{R}(a^k) = \frac{1}{1+\mathrm{e}^{2(k+1)}} \cdot \prod_{t=0}^{k-1} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}$$

$$\sum_{k\in\mathbb{N}} \mathcal{R}(a^k) = 1 - \prod_{t\in\mathbb{N}} \frac{1}{1+\mathrm{e}^{-2(t+1)}} = 1 - \frac{2}{(-1;\mathrm{e}^{-2})_\infty} < 0.14$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 2: TM M halts after exactly ℓ steps

$$s_t^w = \begin{pmatrix} t \overset{?}{>} \ell \\ \sigma\langle 3(t-\ell-1)\rangle \\ t+1 \\ \vdots \end{pmatrix}$$

$$\text{out}_t^w(\$) = \begin{cases} \frac{1}{1+e^{2(t+1)}} & \text{if } t \leq \ell \\ \frac{e^{t-3\ell-5}}{1+e^{t-3\ell-5}} & \text{otherwise} \end{cases}$$

Decidability

Prediction matrix $E = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ $\begin{matrix} \$ \\ \\ a \end{matrix}$

- Case 2: TM M halts after exactly ℓ steps

$$s_t^w = \begin{pmatrix} t \overset{?}{>} \ell \\ \sigma\langle 3(t-\ell-1)\rangle \\ t+1 \\ \vdots \end{pmatrix} \qquad \text{out}_t^w(\$) = \begin{cases} \frac{1}{1+e^{2(t+1)}} & \text{if } t \leq \ell \\ \frac{e^{t-3\ell-5}}{1+e^{t-3\ell-5}} & \text{otherwise} \end{cases}$$

Hence $\mathcal{R}$ is consistent because

$$\sum_{k \in \mathbb{N}} \mathcal{R}(a^k) = 1$$

(Appendix Lemma B)

Decidability

Notes

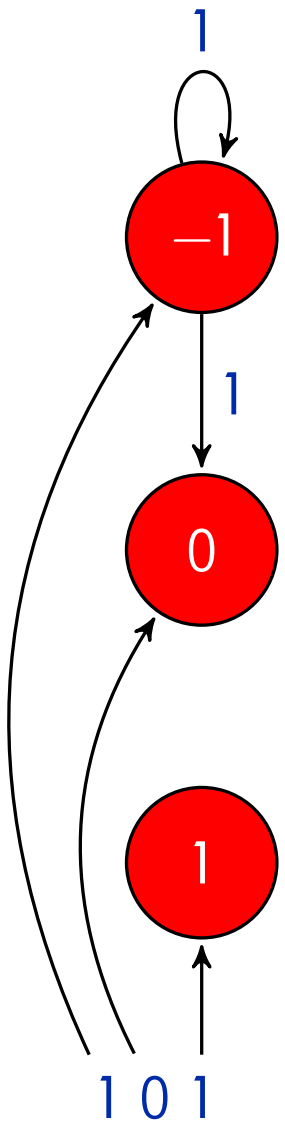
- Consistency undecidable by previous theorem
- Backpropagation-through-time (BPTT) might return inconsistent RNN

	Best path	Best string
RNN	**Undecidable**	
Consistent RNN	**NP-c**†	
Det. WA/PCFG	**P**	
WA/PCFG		**NP-c**

† restricted to solutions of polynomial length

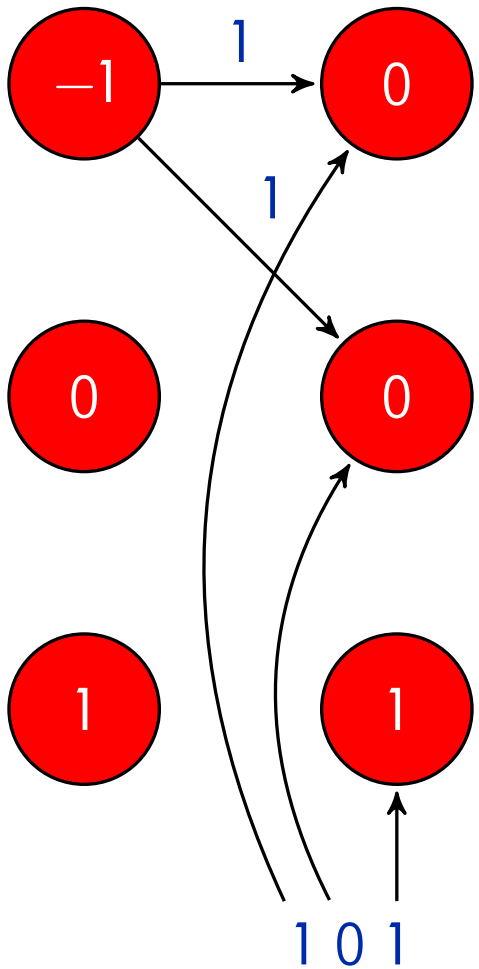
Training

Unrolling



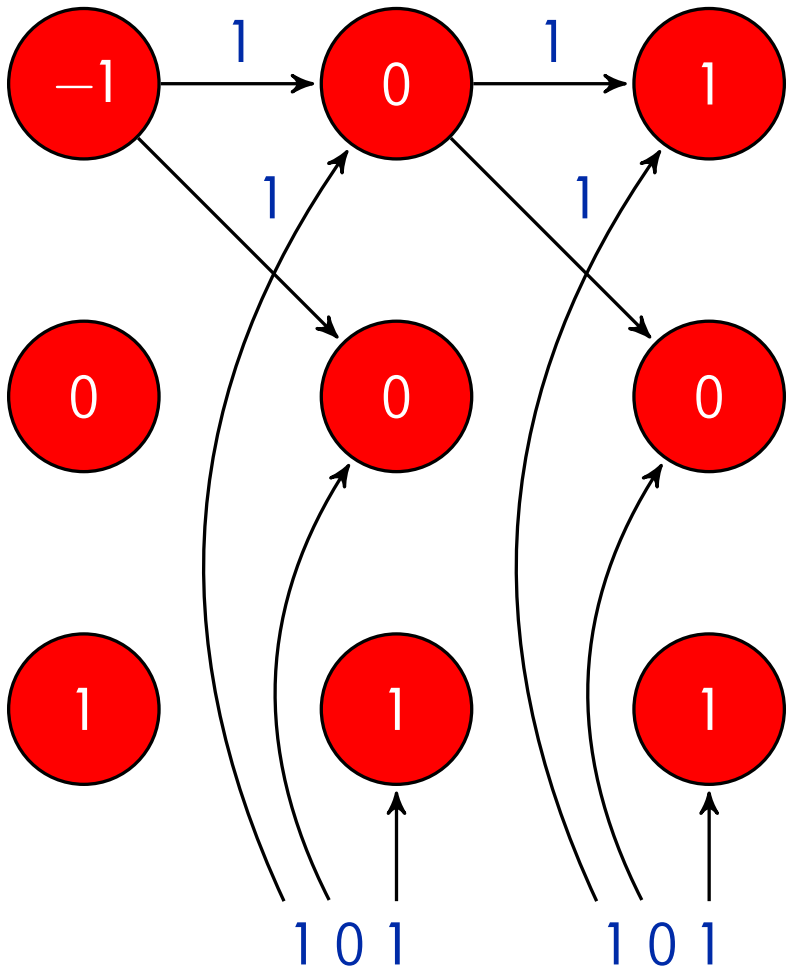
Training

Unrolling



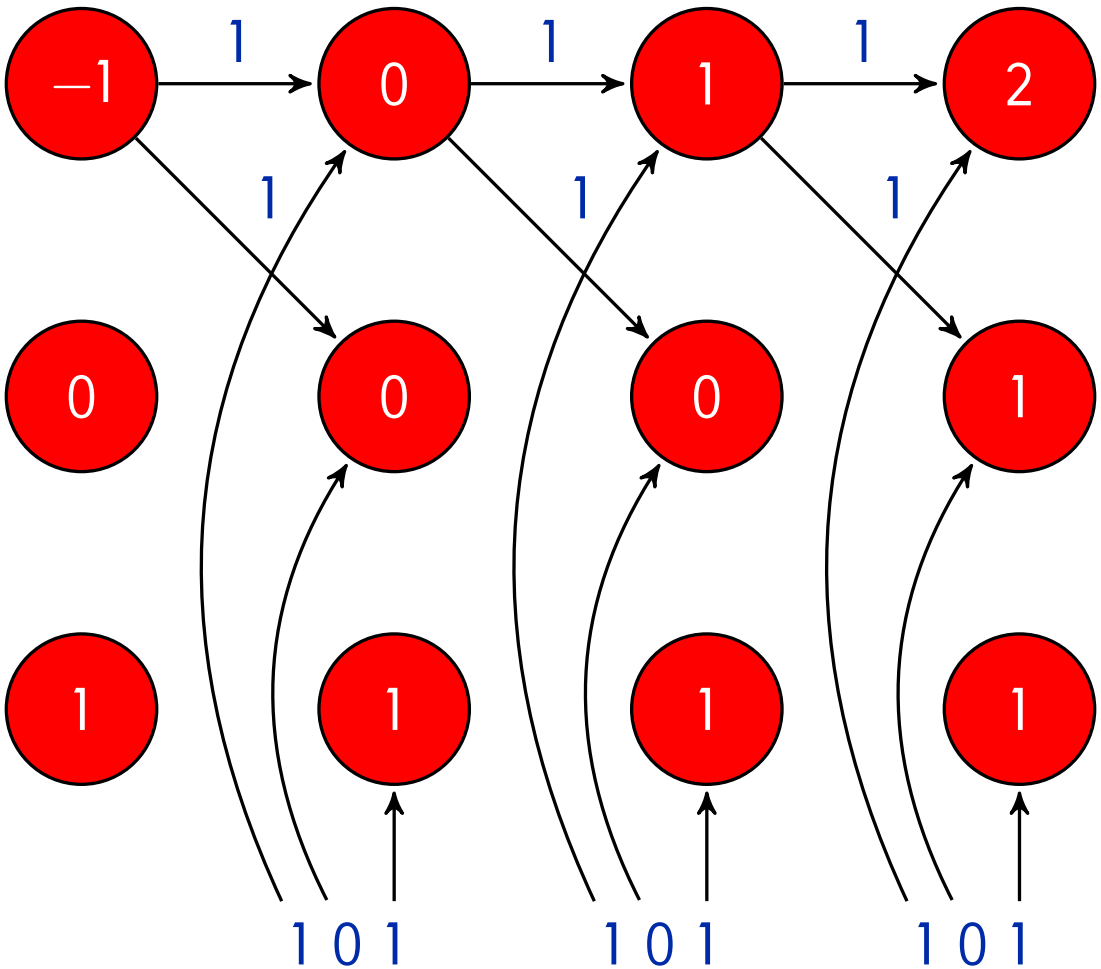
Training

Unrolling



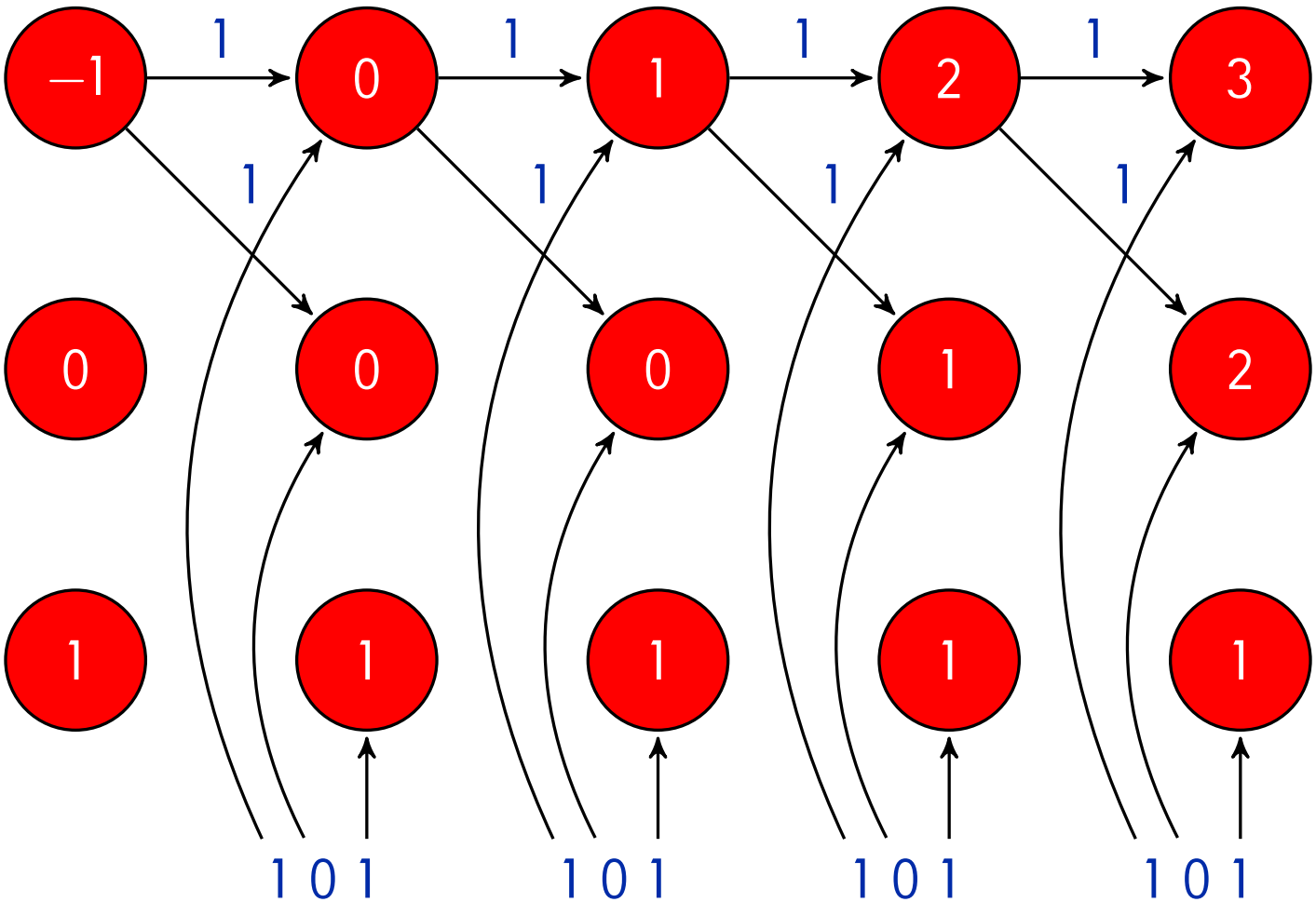
Training

Unrolling



Training

Unrolling



Training

Loss function

- Measures errors currently made
- Cross-entropy loss for predictions p

Training

Loss function

- Measures errors currently made
- Cross-entropy loss for predictions p
  - Next letter given input w with reference a

$$L(w, a, p) = -\log p(w, a)$$

Training

Loss function

- Measures errors currently made
- Cross-entropy loss for predictions p
  - Next letter given input w with reference a

$$L(w, a, p) = -\log p(w, a)$$

  - Input $w \in \mathcal{C}$

$$L(w, p) = \operatorname*{Avg}_{1 \leq i \leq |w|} L\big(w[1] \cdots w[i-1], w[i], p\big)$$

Training

Loss function

- Measures errors currently made
- Cross-entropy loss for predictions p
  - Next letter given input w with reference a

$$L(w, a, p) = -\log p(w, a)$$

  - Input $w \in \mathcal{C}$

$$L(w, p) = \operatorname*{Avg}_{1 \leq i \leq |w|} L\big(w[1] \cdots w[i-1], w[i], p\big)$$

  - Language $\mathcal{C}$

$$L(\mathcal{C}, p) = \operatorname*{Avg}_{w \in \mathcal{C}} L(w, p)$$

Training

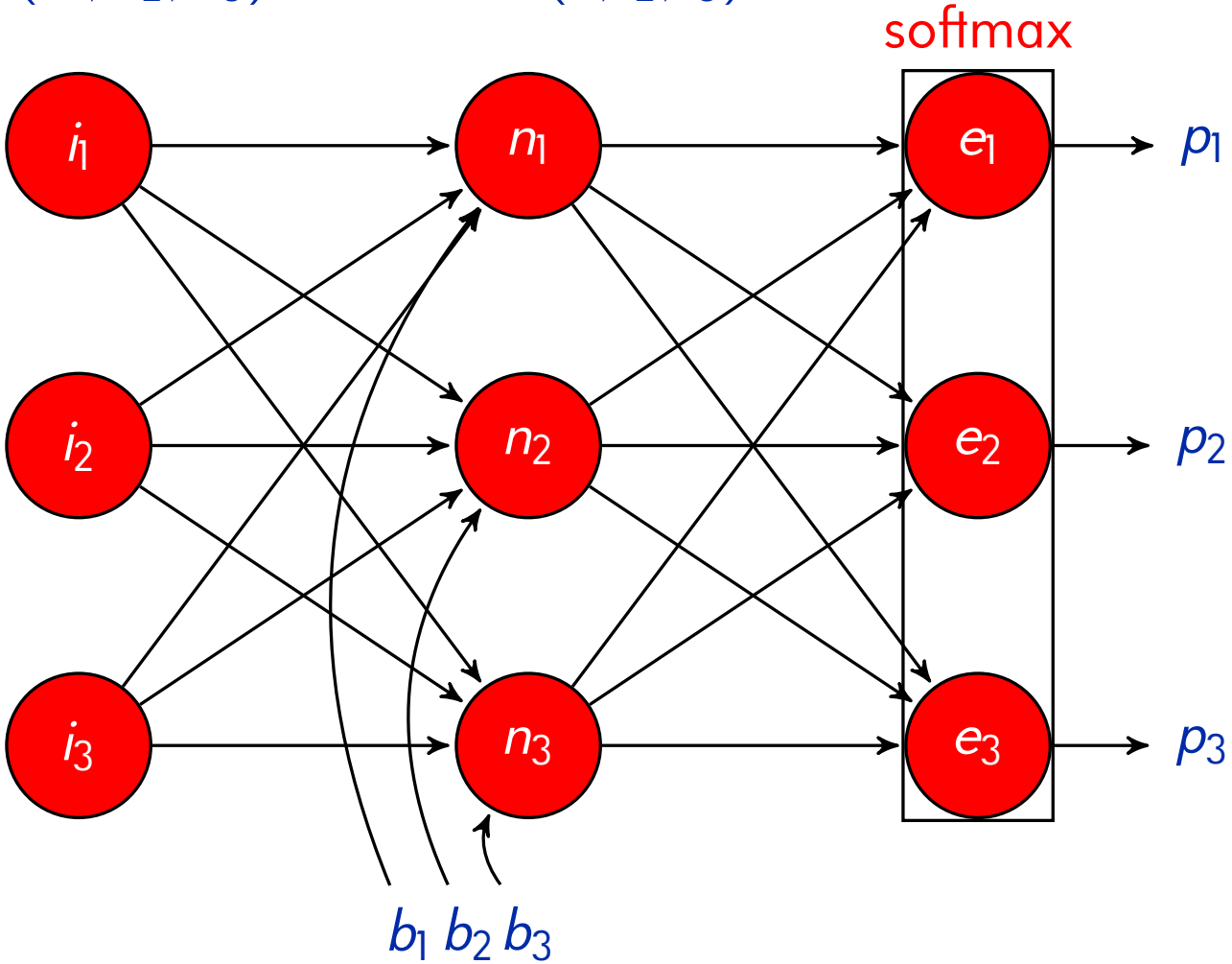
Feed-forward network

- First letter, $|\Sigma| = 2$ letters ($|\Sigma_\$| = 3$ outputs), $|N| = 3$ neurons
- $\text{inp}_\$ = (b_1, b_2, b_3)$ and $s_{-1} = (i_1, i_2, i_3)$

Training

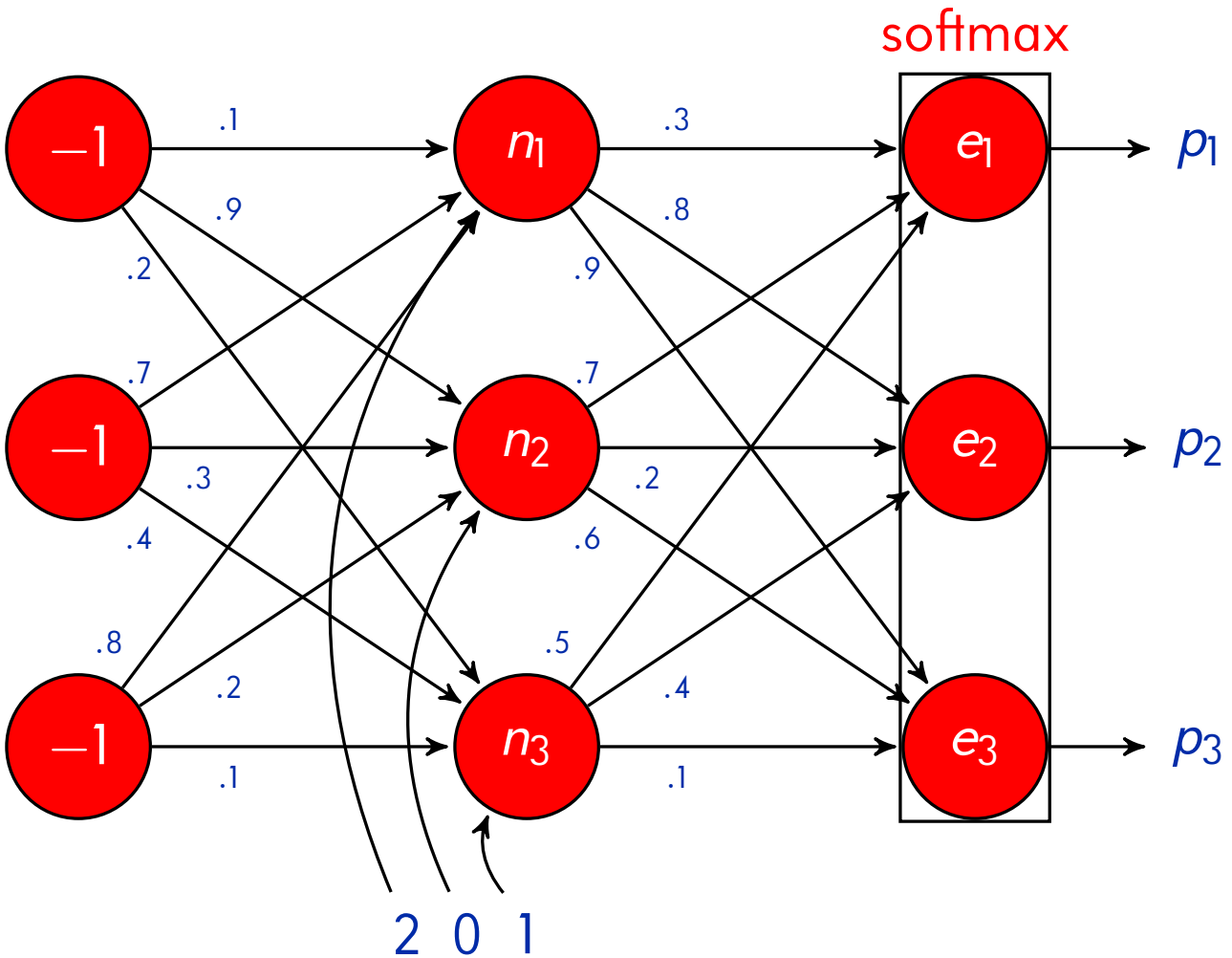
Feed-forward network

- First letter, $|\Sigma| = 2$ letters ($|\Sigma_\$| = 3$ outputs), $|N| = 3$ neurons
- $\text{inp}_\$ = (b_1, b_2, b_3)$ and $s_{-1} = (i_1, i_2, i_3)$



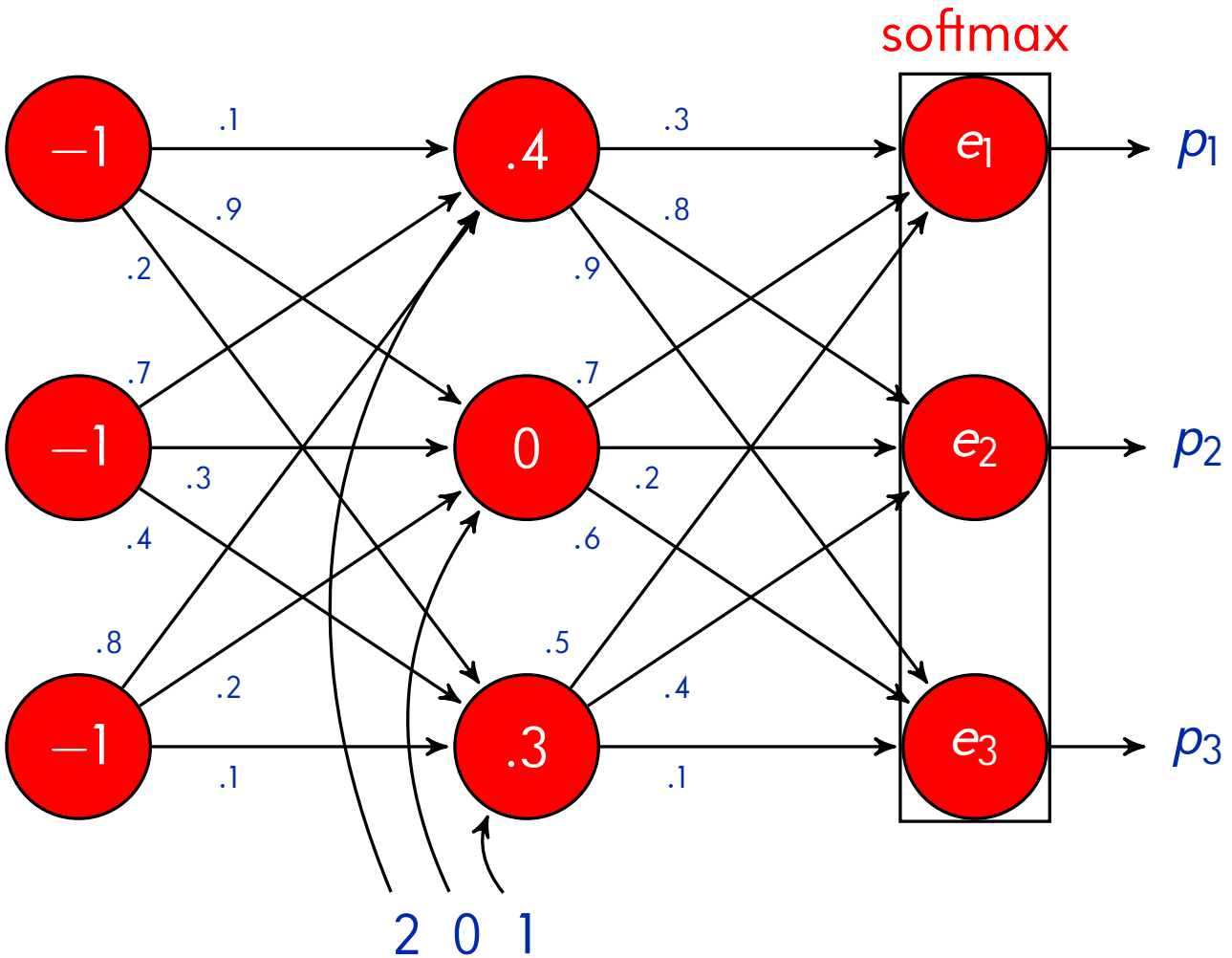
Training

Forward pass



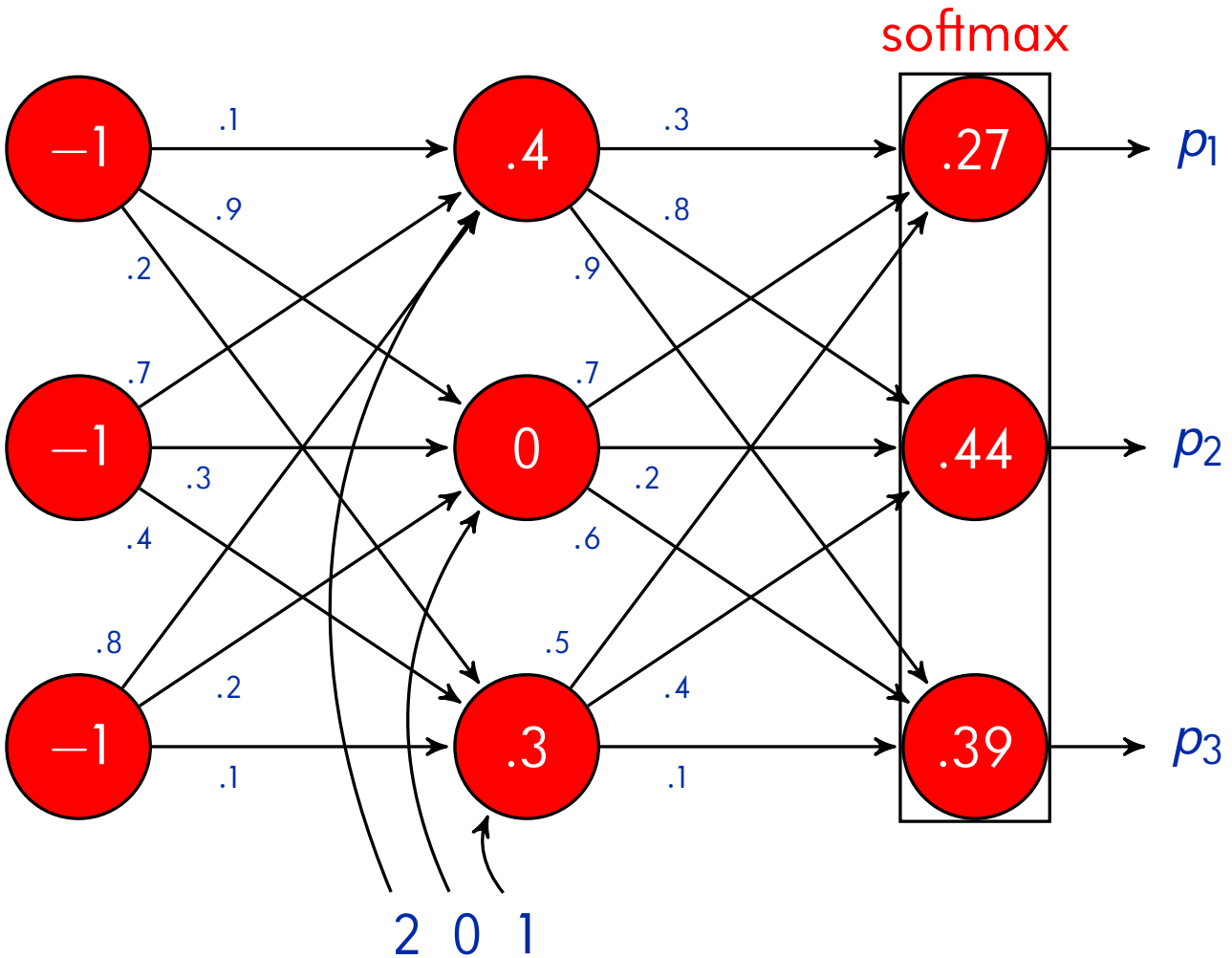
Training

Forward pass



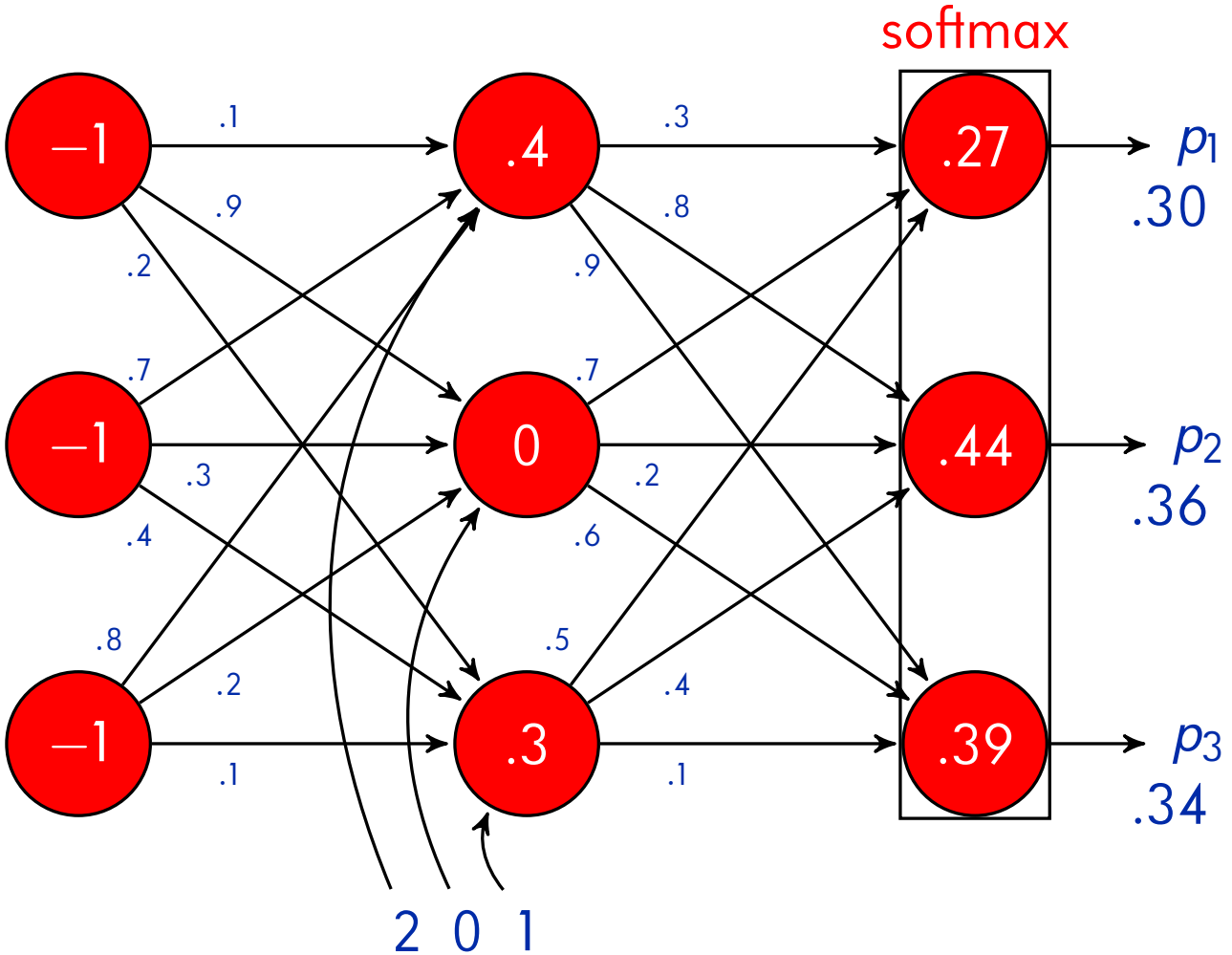
Training

Forward pass



Training

Forward pass



Training

Calculation of loss

- Obtain output probabilities $(p_1, p_2, p_3) \approx (.30, .36, .34)$
- Loss $L = -\log(p_3) \approx 1.08$ for correct output corresponding to p_3

Training

Calculation of loss

- Obtain output probabilities $(p_1, p_2, p_3) \approx (.30, .36, .34)$
- Loss $L = -\log(p_3) \approx 1.08$ for correct output corresponding to p_3

Assigning blame and computing changes

- Want lower loss, so calculate partial derivatives of loss function

Training

Calculation of loss

- Obtain output probabilities $(p_1, p_2, p_3) \approx (.30, .36, .34)$
- Loss $L = -\log(p_3) \approx 1.08$ for correct output corresponding to p_3

Assigning blame and computing changes

- Want lower loss, so calculate partial derivatives of loss function
- Treat all edge weights $\mathrm{wt}(\cdot, \cdot)$ as variables (here: entries of wt and E)
- Calculate $\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)}$ with $L = -\log p_3$

Training

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3} \frac{\partial p_3}{\partial e_3} \frac{\partial e_3}{\partial z_3} \frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)}$$

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3} \frac{\partial p_3}{\partial e_3} \frac{\partial e_3}{\partial z_3} \frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_3} = -p_3^{-1}$

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3} \frac{\partial p_3}{\partial e_3} \frac{\partial e_3}{\partial z_3} \frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_3} = -p_3^{-1}$
- $\frac{\partial p_3}{\partial e_3} = \frac{e^{e_3}(e^{e_1}+e^{e_2}+e^{e_3}) - e^{e_3} \cdot e^{e_3}}{(e^{e_1}+e^{e_2}+e^{e_3})^2} = \frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}} - \left(\frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}}\right)^2 = p_3(1-p_3)$

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3} \frac{\partial p_3}{\partial e_3} \frac{\partial e_3}{\partial z_3} \frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_3} = -p_3^{-1}$
- $\frac{\partial p_3}{\partial e_3} = \frac{e^{e_3}(e^{e_1}+e^{e_2}+e^{e_3}) - e^{e_3} \cdot e^{e_3}}{(e^{e_1}+e^{e_2}+e^{e_3})^2} = \frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}} - \left(\frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}}\right)^2 = p_3(1-p_3)$
- $\frac{\partial e_3}{\partial z_3} = 1$ if $z_3 > 0$ and 0 otherwise (well almost)

Training

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3} \frac{\partial p_3}{\partial e_3} \frac{\partial e_3}{\partial z_3} \frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_3} = -p_3^{-1}$
- $\frac{\partial p_3}{\partial e_3} = \frac{e^{e_3}(e^{e_1}+e^{e_2}+e^{e_3}) - e^{e_3} \cdot e^{e_3}}{(e^{e_1}+e^{e_2}+e^{e_3})^2} = \frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}} - \left(\frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}}\right)^2 = p_3(1-p_3)$
- $\frac{\partial e_3}{\partial z_3} = 1$ if $z_3 > 0$ and 0 otherwise (well almost)
- $\frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)} = n_1$

Training

Computing derivatives

$$L = -\log p_3 \qquad p_3 = \frac{e^{e_3}}{e^{e_1} + e^{e_2} + e^{e_3}}$$

$$e_3 = \max(0, z_3) \qquad z_3 = n_1 \cdot \mathrm{wt}(n_1, e_3) + n_2 \cdot \mathrm{wt}(n_2, e_3) + n_3 \cdot \mathrm{wt}(n_3, e_3)$$

- Chain rule

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{\partial L}{\partial p_3}\frac{\partial p_3}{\partial e_3}\frac{\partial e_3}{\partial z_3}\frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)} = -\frac{\cancel{p_3}}{\cancel{p_3}}(1 - p_3)n_1 = n_1(p_3 - 1)$$

- $\frac{\partial L}{\partial p_3} = -p_3^{-1}$
- $\frac{\partial p_3}{\partial e_3} = \frac{e^{e_3}(e^{e_1}+e^{e_2}+e^{e_3}) - e^{e_3}\cdot e^{e_3}}{(e^{e_1}+e^{e_2}+e^{e_3})^2} = \frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}} - \left(\frac{e^{e_3}}{e^{e_1}+e^{e_2}+e^{e_3}}\right)^2 = p_3(1-p_3)$
- $\frac{\partial e_3}{\partial z_3} = 1$ if $z_3 > 0$ and 0 otherwise (well almost)
- $\frac{\partial z_3}{\partial \mathrm{wt}(n_1, e_3)} = n_1$

Training

Multiple inputs

- Superscript in parentheses indicating input
  e.g. $n_3^{(i)}$ for n_3 of i-th input
- Indicator bit $t_k^{(i)} \in \{0,1\}$ indicating
  whether desired output symbol for i-th input corresponds to p_k

Training

Computing derivatives for ℓ inputs

$$L = -\frac{1}{\ell} \sum_{i=1}^{\ell} \sum_{k=1}^{3} t_k^{(i)} \log p_k^{(i)} \qquad p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}}$$

$$e_m^{(i)} = \max(0, z_m^{(i)}) \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

Training

Computing derivatives for ℓ inputs

$$L = -\frac{1}{\ell} \sum_{i=1}^{\ell} \sum_{k=1}^{3} t_k^{(i)} \log p_k^{(i)} \qquad p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}}$$

$$e_m^{(i)} = \max(0, z_m^{(i)}) \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

- Chain rule

$$\frac{\partial L}{\partial \,\mathrm{wt}(n_1, e_3)} = \sum_{i=1}^{\ell} \sum_{k=1}^{3} \sum_{m=1}^{3} \frac{\partial L}{\partial p_k^{(i)}} \frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} \frac{\partial e_m^{(i)}}{\partial z_m^{(i)}} \frac{\partial z_m^{(i)}}{\partial \,\mathrm{wt}(n_1, e_3)}$$

Training

Computing derivatives for ℓ inputs

$$L = -\frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{k=1}^{3} t_k^{(i)} \log p_k^{(i)} \qquad p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}}$$

$$e_m^{(i)} = \max(0, z_m^{(i)}) \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

- Chain rule

$$\frac{\partial L}{\partial \,\mathrm{wt}(n_1, e_3)} = \sum_{i=1}^{\ell}\sum_{k=1}^{3}\sum_{m=1}^{3} \frac{\partial L}{\partial p_k^{(i)}} \frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} \frac{\partial e_m^{(i)}}{\partial z_m^{(i)}} \frac{\partial z_m^{(i)}}{\partial \,\mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_k^{(i)}} = -\frac{t_k^{(i)}}{\ell p_k^{(i)}}$

Training

Computing derivatives for ℓ inputs

$$L = -\frac{1}{\ell}\sum_{i=1}^{\ell}\sum_{k=1}^{3} t_k^{(i)} \log p_k^{(i)} \qquad p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}}$$

$$e_m^{(i)} = \max(0, z_m^{(i)}) \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

- Chain rule

$$\frac{\partial L}{\partial \,\mathrm{wt}(n_1, e_3)} = \sum_{i=1}^{\ell}\sum_{k=1}^{3}\sum_{m=1}^{3} \frac{\partial L}{\partial p_k^{(i)}} \frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} \frac{\partial e_m^{(i)}}{\partial z_m^{(i)}} \frac{\partial z_m^{(i)}}{\partial \,\mathrm{wt}(n_1, e_3)}$$

- $\frac{\partial L}{\partial p_k^{(i)}} = -\frac{t_k^{(i)}}{\ell p_k^{(i)}}$
- $\frac{\partial e_m^{(i)}}{\partial z_m^{(i)}} = 1$ if $z_m^{(i)} > 0$ and 0 otherwise (well almost)

Training

Computing derivatives for ℓ inputs

$$p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}} \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

$$\frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} = \begin{cases} p_k^{(i)}(1 - p_k^{(i)}) & \text{if } k = m \\ -p_k^{(i)} p_m^{(i)} & \text{otherwise} \end{cases}$$

$$\frac{\partial z_m^{(i)}}{\partial \, \mathrm{wt}(n_1, e_3)} = \begin{cases} n_1^{(i)} & \text{if } m = 3 \\ 0 & \text{otherwise} \end{cases}$$

Training

Computing derivatives for ℓ inputs

$$p_k^{(i)} = \frac{e^{e_k^{(i)}}}{e^{e_1^{(i)}} + e^{e_2^{(i)}} + e^{e_3^{(i)}}} \qquad z_m^{(i)} = \sum_{h=1}^{3} n_h^{(i)} \cdot \mathrm{wt}(n_h, e_m)$$

$$\frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} = \begin{cases} p_k^{(i)}(1 - p_k^{(i)}) & \text{if } k = m \\ -p_k^{(i)} p_m^{(i)} & \text{otherwise} \end{cases}$$

$$\frac{\partial z_m^{(i)}}{\partial \mathrm{wt}(n_1, e_3)} = \begin{cases} n_1^{(i)} & \text{if } m = 3 \\ 0 & \text{otherwise} \end{cases}$$

Thus

$$\frac{\partial L}{\partial \mathrm{wt}(n_1, e_3)} = \frac{1}{\ell} \sum_{i \in \{1,\ldots,\ell\}, z_3^{(i)} > 0} (p_3^{(i)} - t_3^{(i)}) n_1^{(i)}$$

(Appendix Lemma C)

Training

Gradient descent

- $f : \mathbb{R}^n \to \mathbb{R}$ defined & differentiable around $a \in \mathbb{R}^n$ decreases fastest going from a in direction of gradient $-\nabla f(a)$

Training

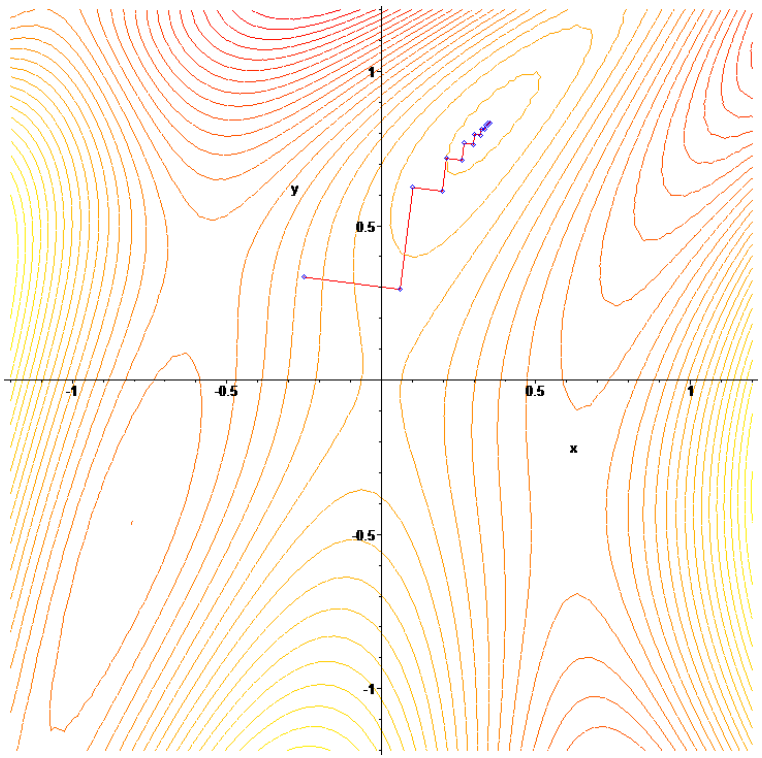
Gradient descent

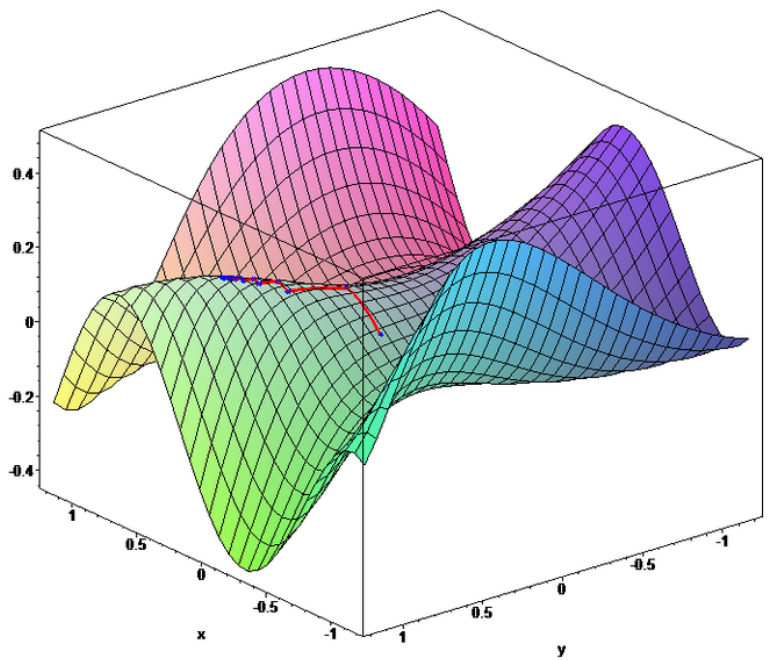
- $f : \mathbb{R}^n \to \mathbb{R}$ defined & differentiable around $a \in \mathbb{R}^n$ decreases fastest going from a in direction of gradient $-\nabla f(a)$
- $a_{n+1} = a_n - \gamma \nabla f(a_n)$ with $\gamma \in \mathbb{R}_+$ small yields $f(a_{n+1}) \leq f(a_n)$

Training

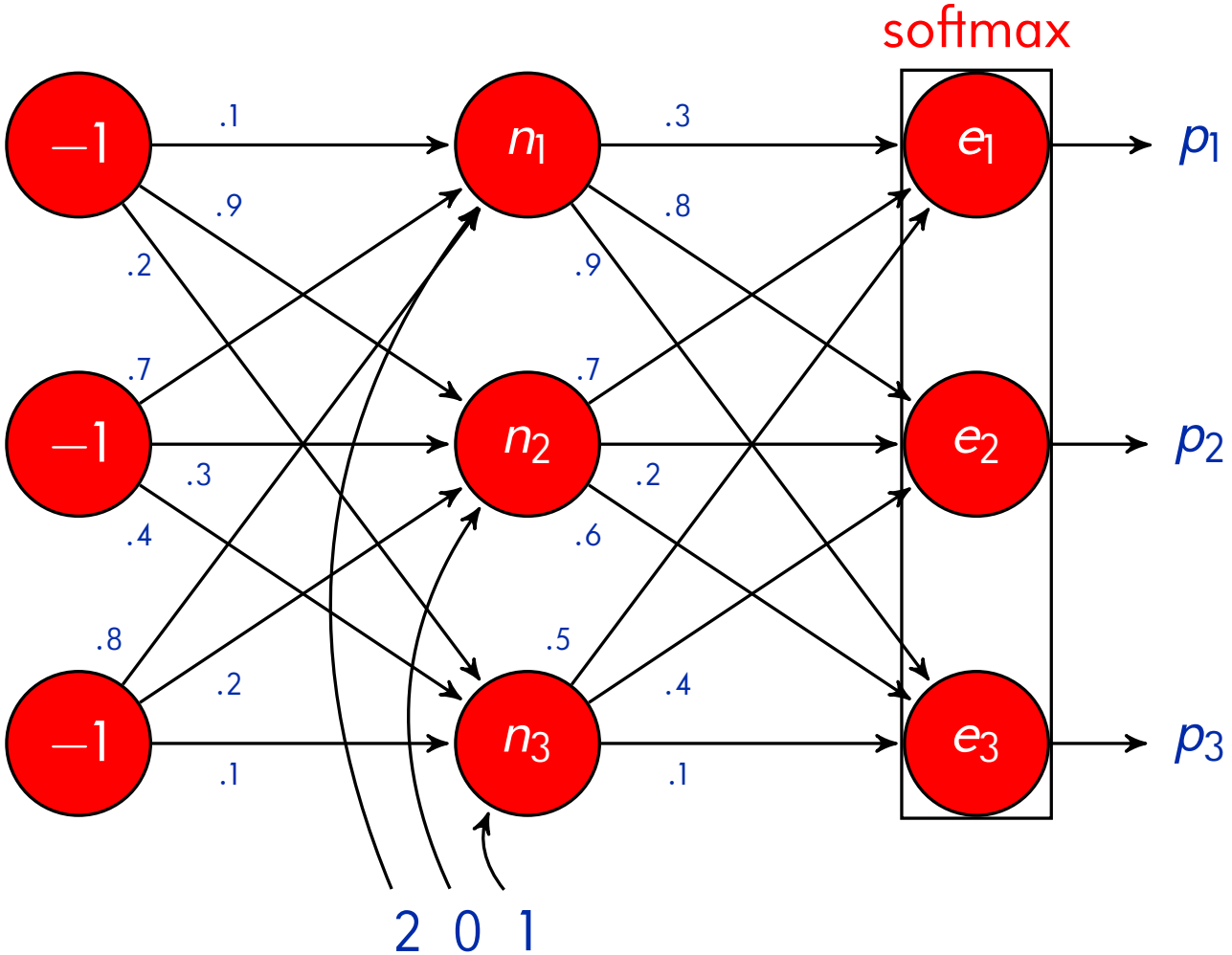
Gradient descent

- $f : \mathbb{R}^n \to \mathbb{R}$ defined & differentiable around $a \in \mathbb{R}^n$ decreases fastest going from a in direction of gradient $-\nabla f(a)$
- $a_{n+1} = a_n - \gamma \nabla f(a_n)$ with $\gamma \in \mathbb{R}_+$ small yields $f(a_{n+1}) \leq f(a_n)$
- Learning rate γ (often varied over time)

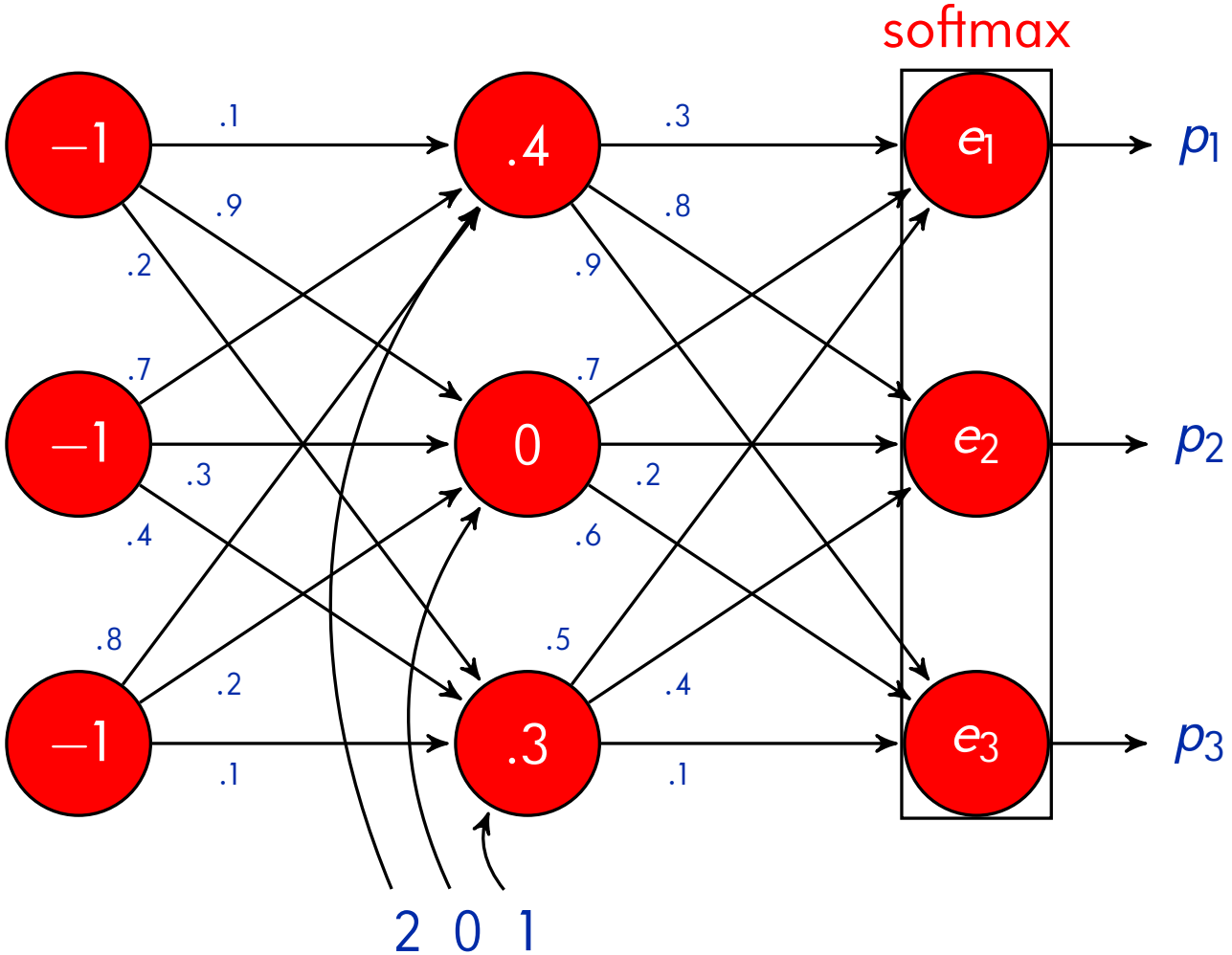




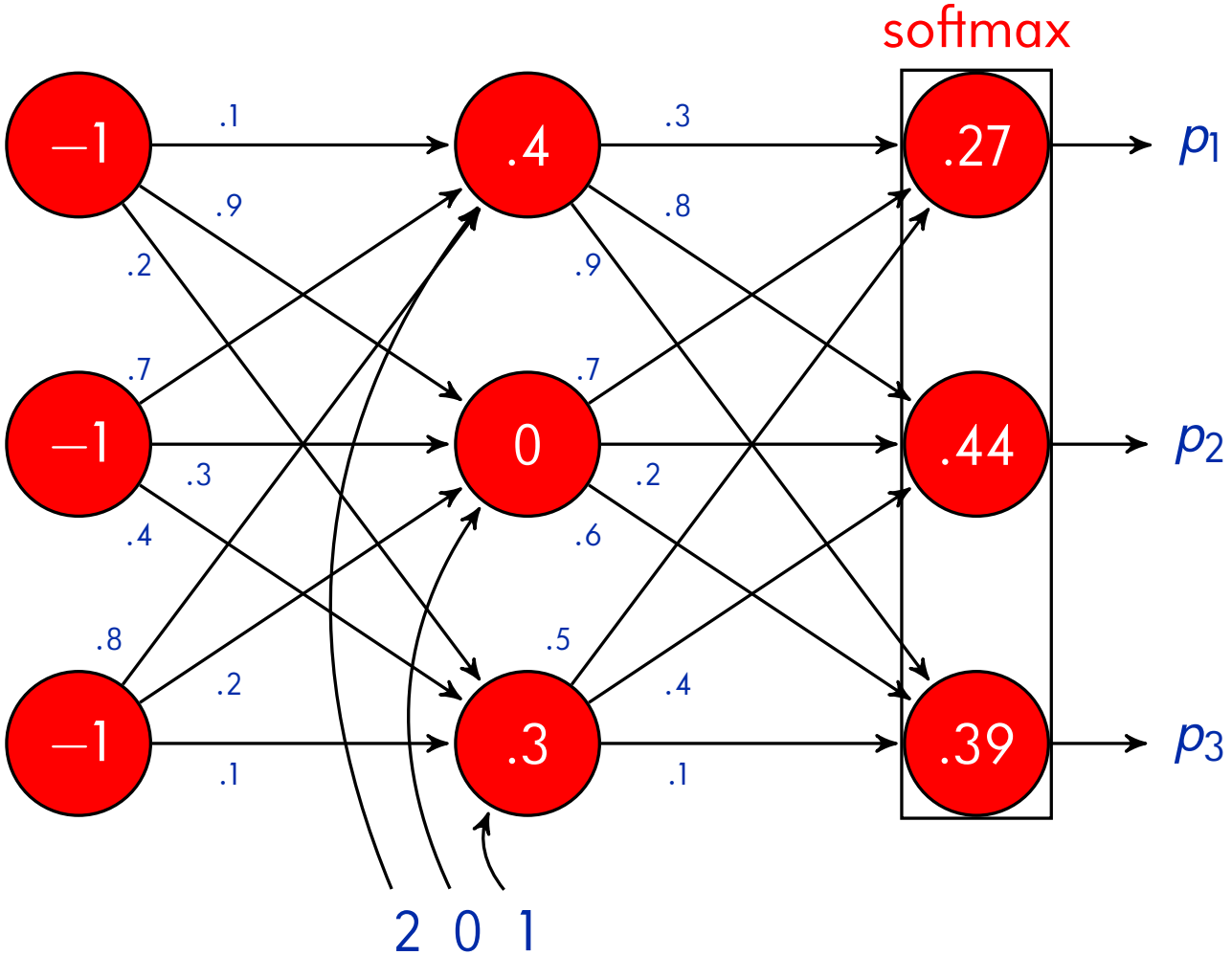
Backpropagation



Backpropagation



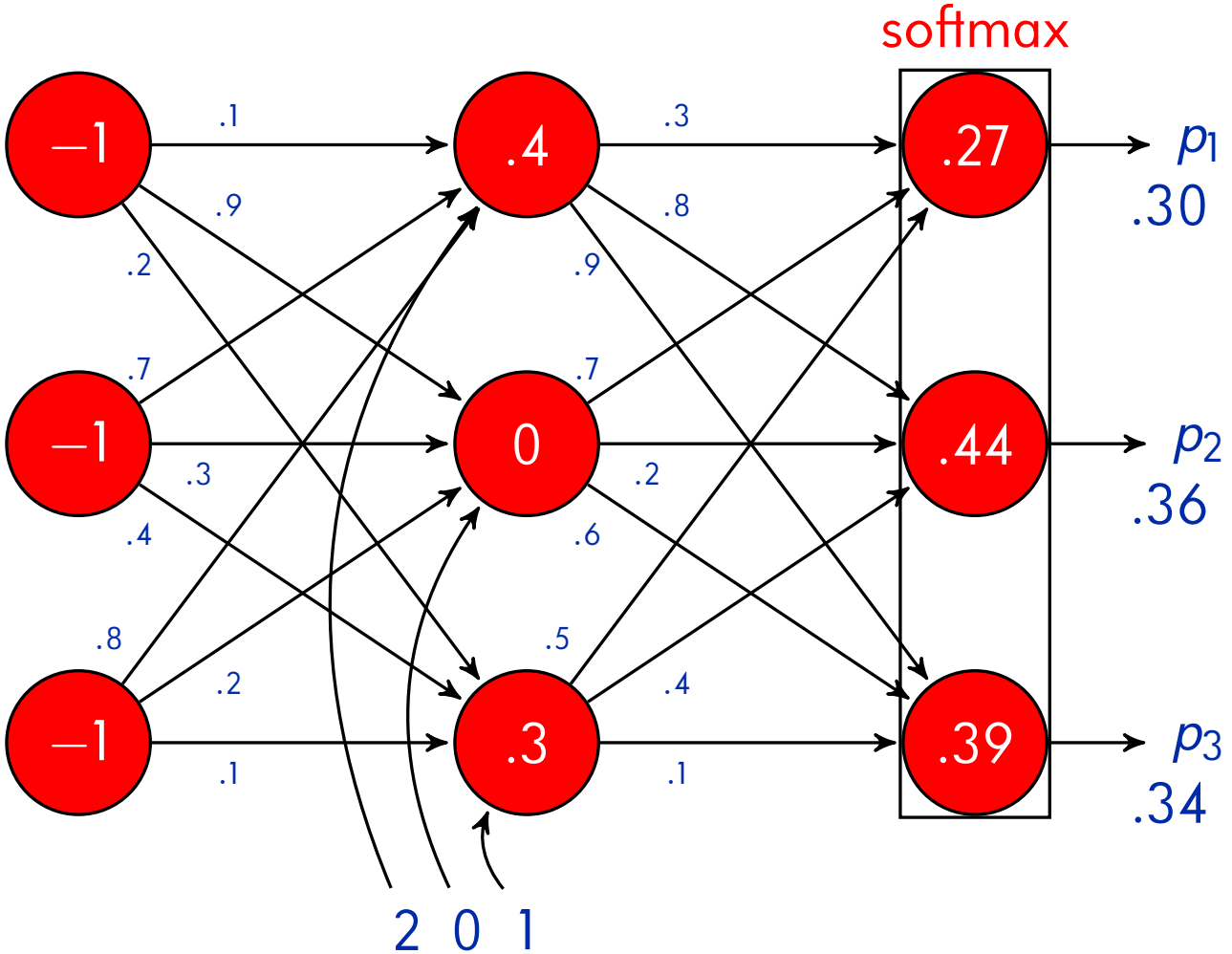
Backpropagation



Training

Backpropagation

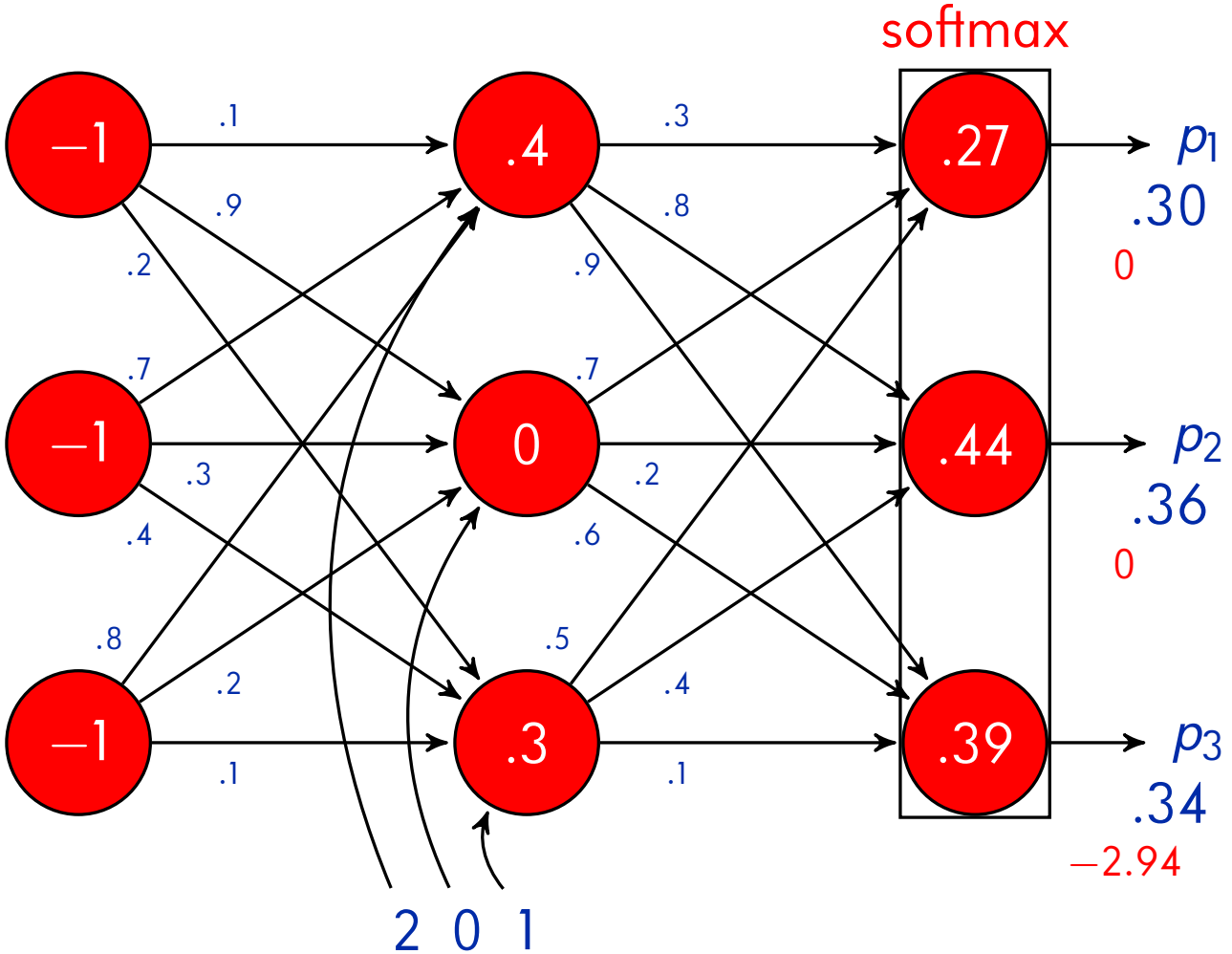
- $\frac{\partial L}{\partial p_1} = \frac{\partial L}{\partial p_2} = 0$ and $\frac{\partial L}{\partial p_3} = -p_3^{-1} \approx -2.94$



Training

Backpropagation

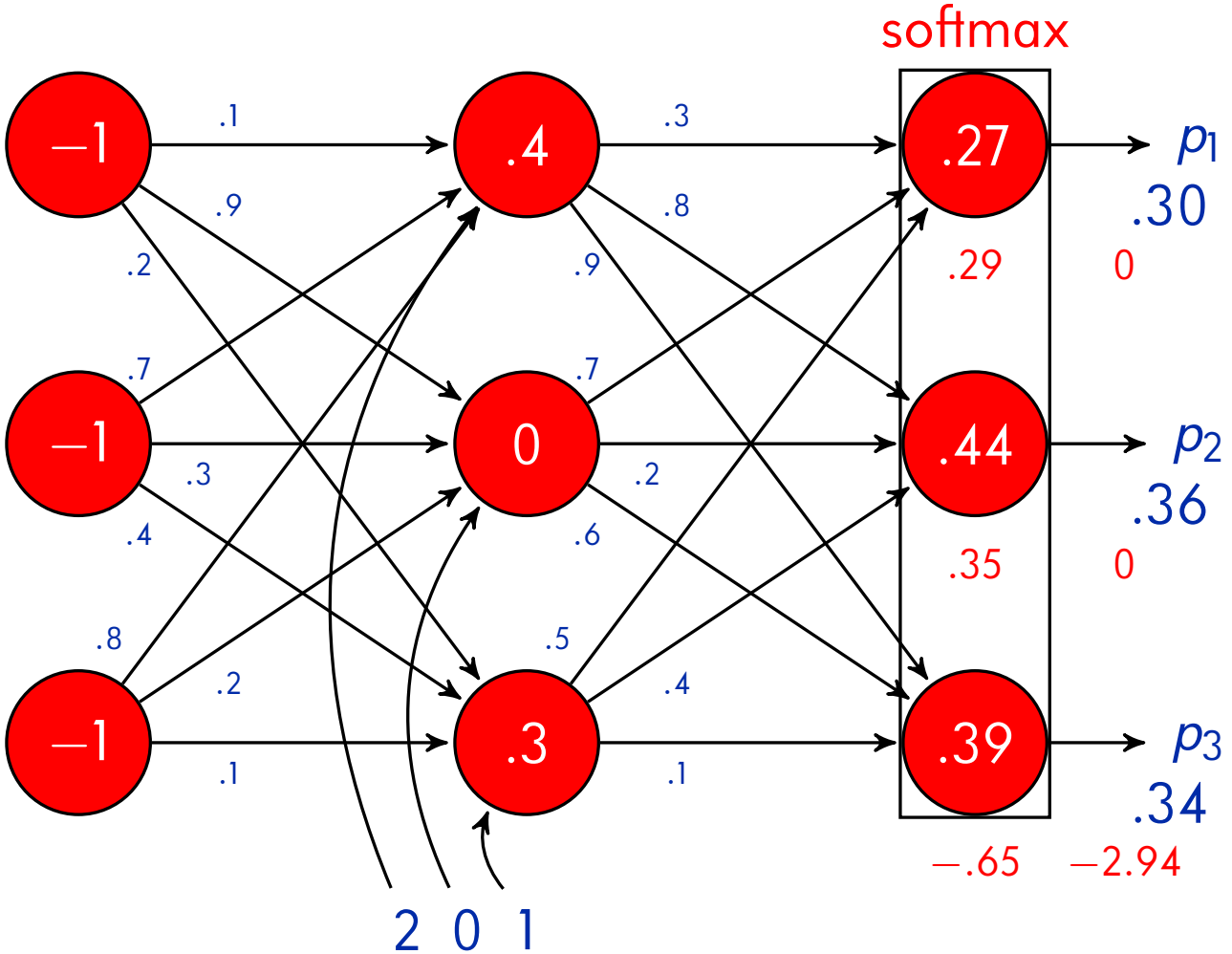
- $\frac{\partial p_3}{\partial e_1} = -p_3 p_1 \approx -0.10$ and $\frac{\partial p_3}{\partial e_3} = p_3(1-p_3) \approx 0.22$



Training

Backpropagation

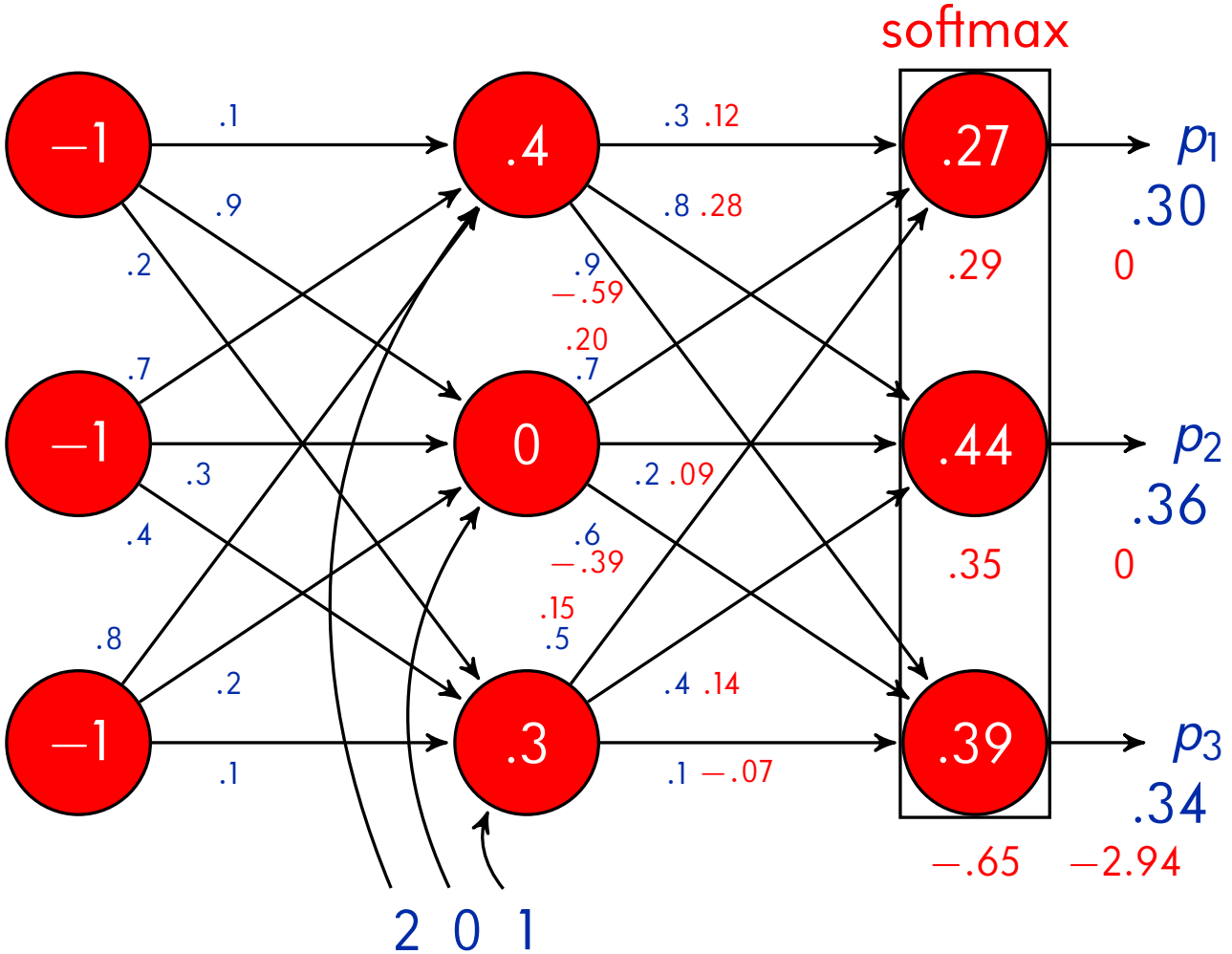
- $\frac{\partial z_m}{\partial \, \mathrm{wt}(n_i, e_m)} = n_i$ for all $i, m \in \{1, 2, 3\}$



Training

Backpropagation

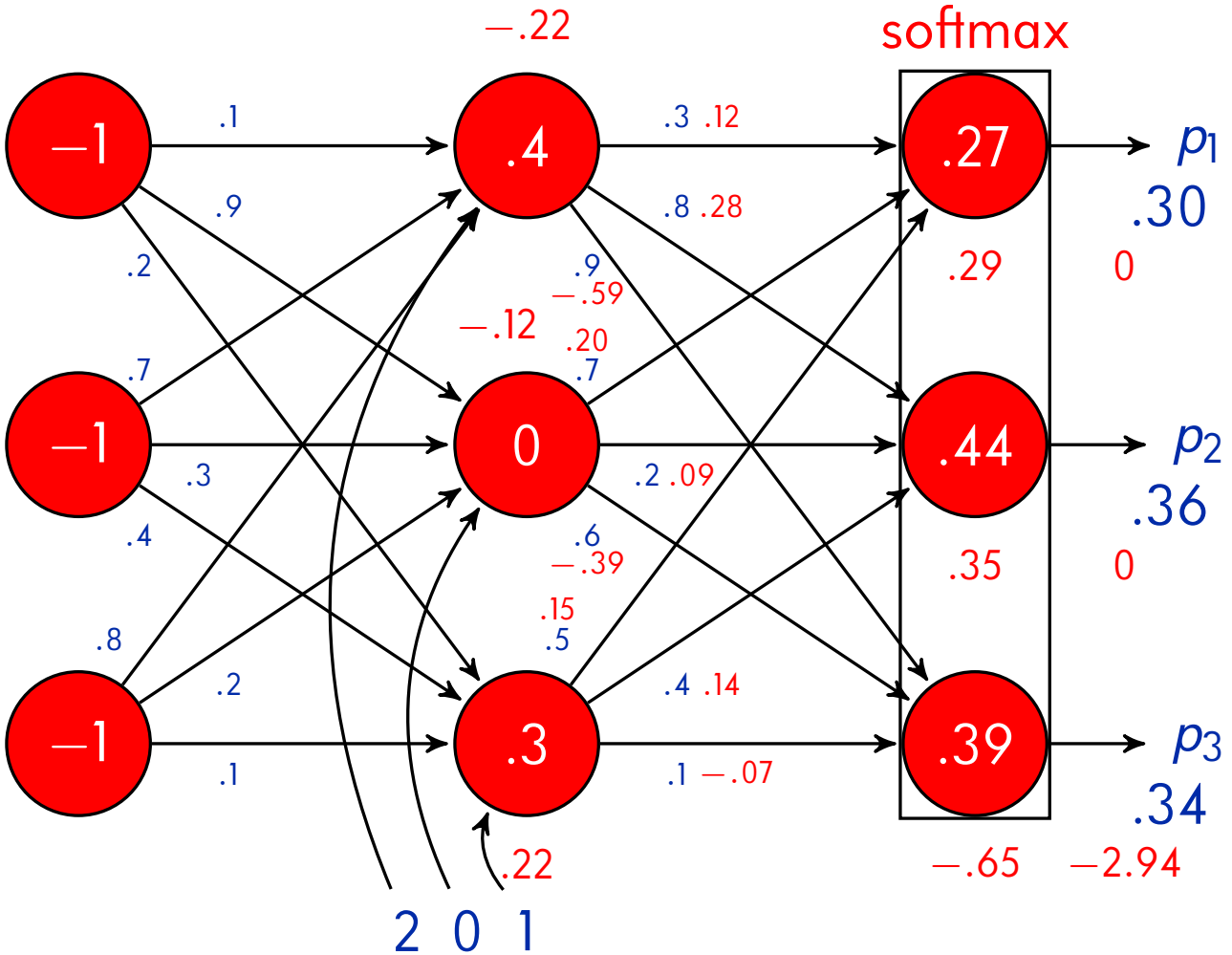
- $\frac{\partial z_m}{\partial n_i} = \text{wt}(n_i, e_m)$ for all $i, m \in \{1, 2, 3\}$



Training

Backpropagation

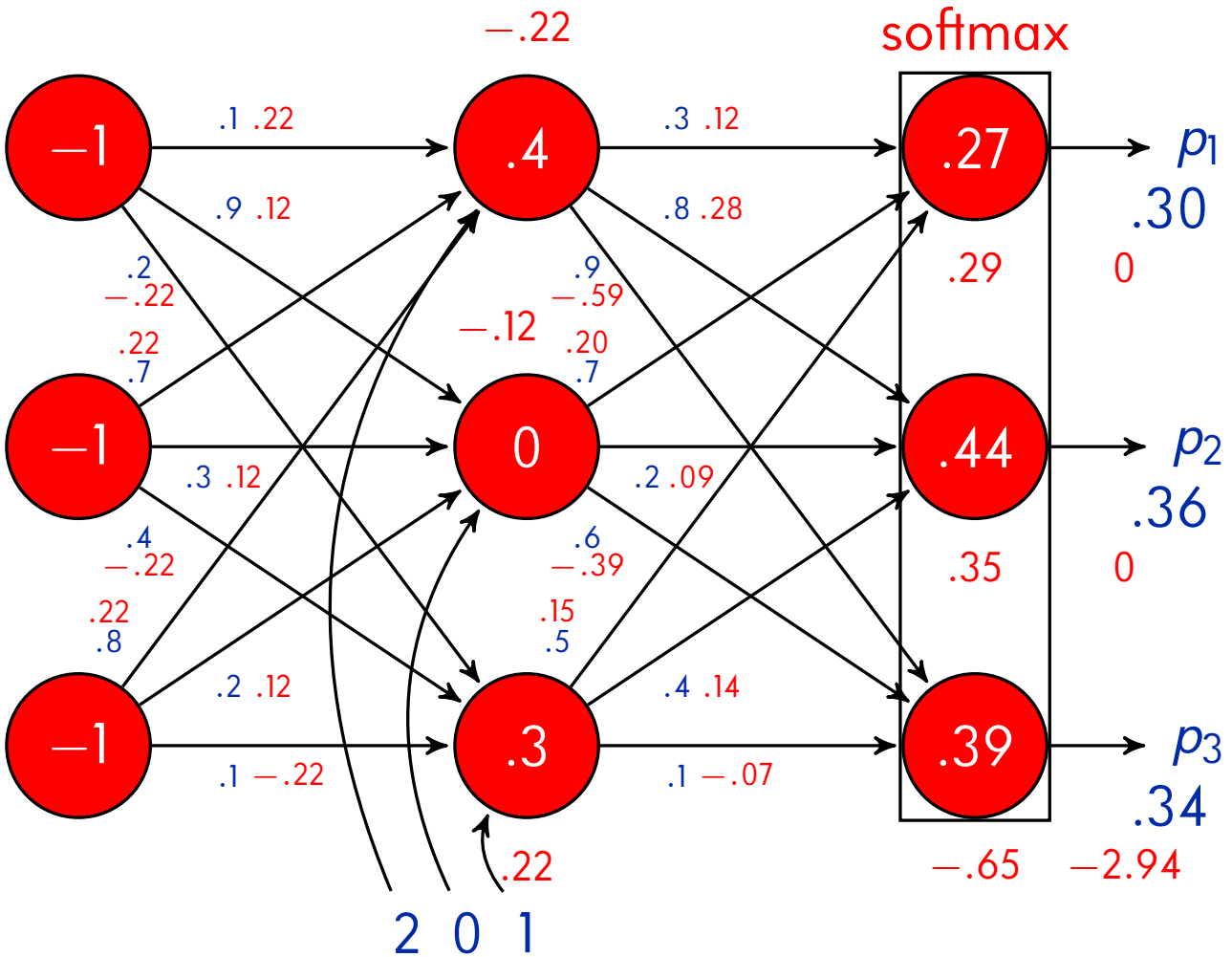
- $\frac{\partial n_i}{\partial \, \mathrm{wt}(i_k, n_i)} = i_k$ for all $i, k \in \{1, 2, 3\}$



Training

Backpropagation

- $\frac{\partial n_i}{\partial \, \mathrm{wt}(i_k, n_i)} = i_k$ for all $i, k \in \{1, 2, 3\}$



Training

Partial derivatives

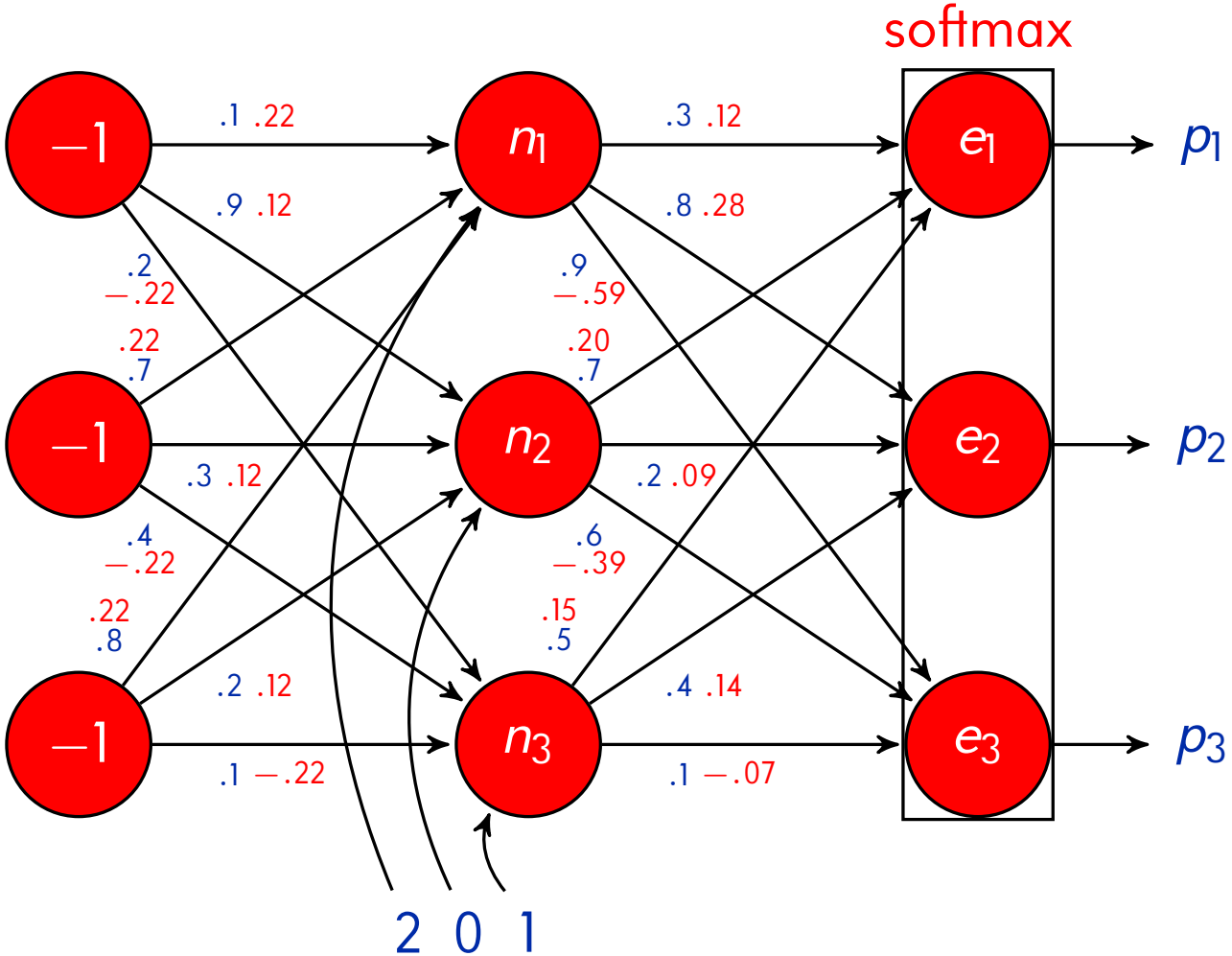
$$\nabla L = \begin{pmatrix} \partial L/\partial \,\mathrm{wt}(i_1, n_1) \\ \partial L/\partial \,\mathrm{wt}(i_1, n_2) \\ \cdots \\ \partial L/\partial \,\mathrm{wt}(i_3, n_3) \\ \partial L/\partial \,\mathrm{wt}(n_1, e_1) \\ \partial L/\partial \,\mathrm{wt}(n_1, e_2) \\ \cdots \\ \partial L/\partial \,\mathrm{wt}(n_3, e_3) \end{pmatrix} \approx \begin{pmatrix} .22 \\ .12 \\ \cdots \\ -.22 \\ .12 \\ .28 \\ \cdots \\ -.07 \end{pmatrix}$$

- Learning rate $\gamma = 1$ (unrealistically large)
- Compute $\mathrm{wt}' = \mathrm{wt} - \gamma \nabla L$

Training

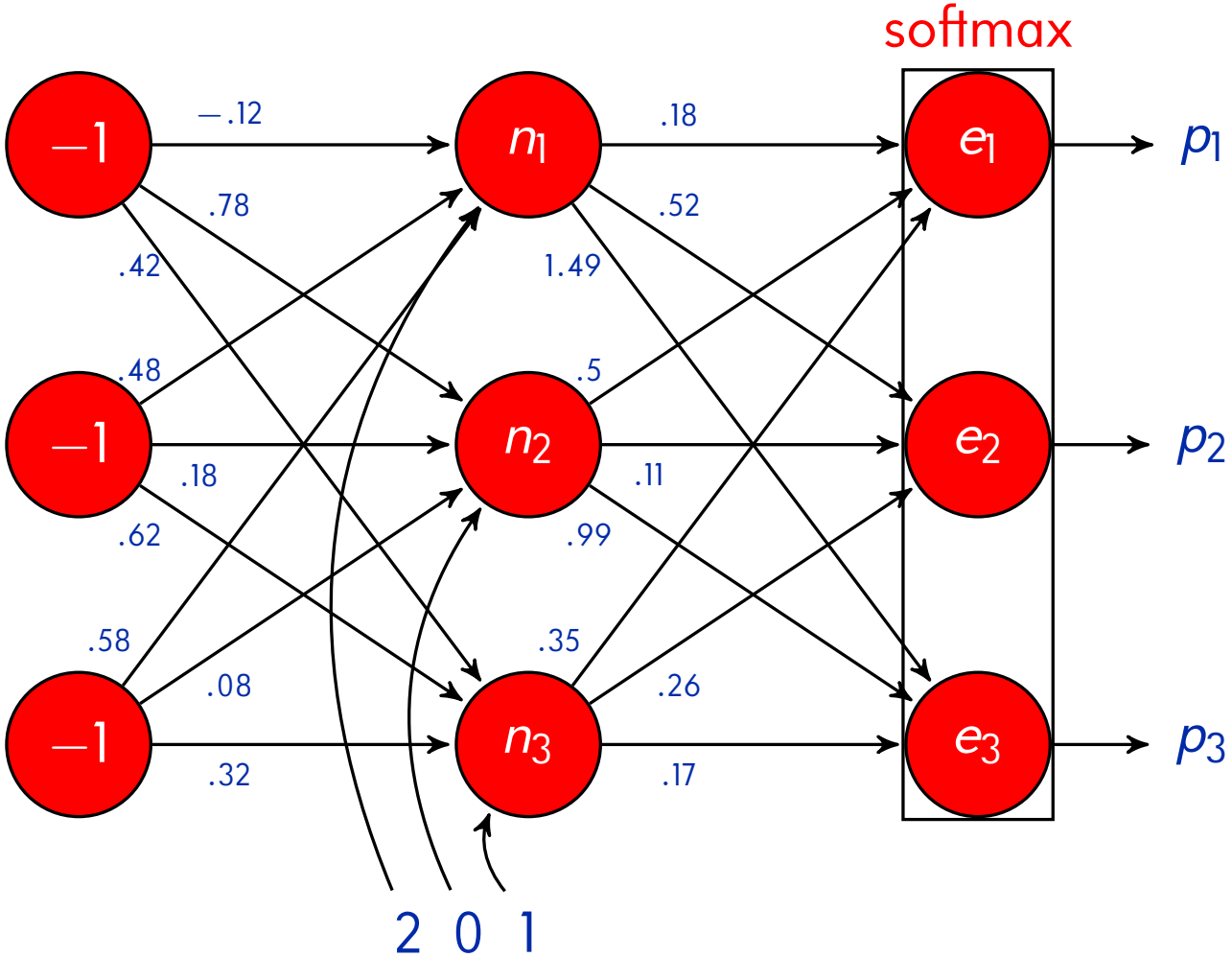
Backpropagation

- Previous loss $-\log(0.34) \approx 1.08$



Backpropagation

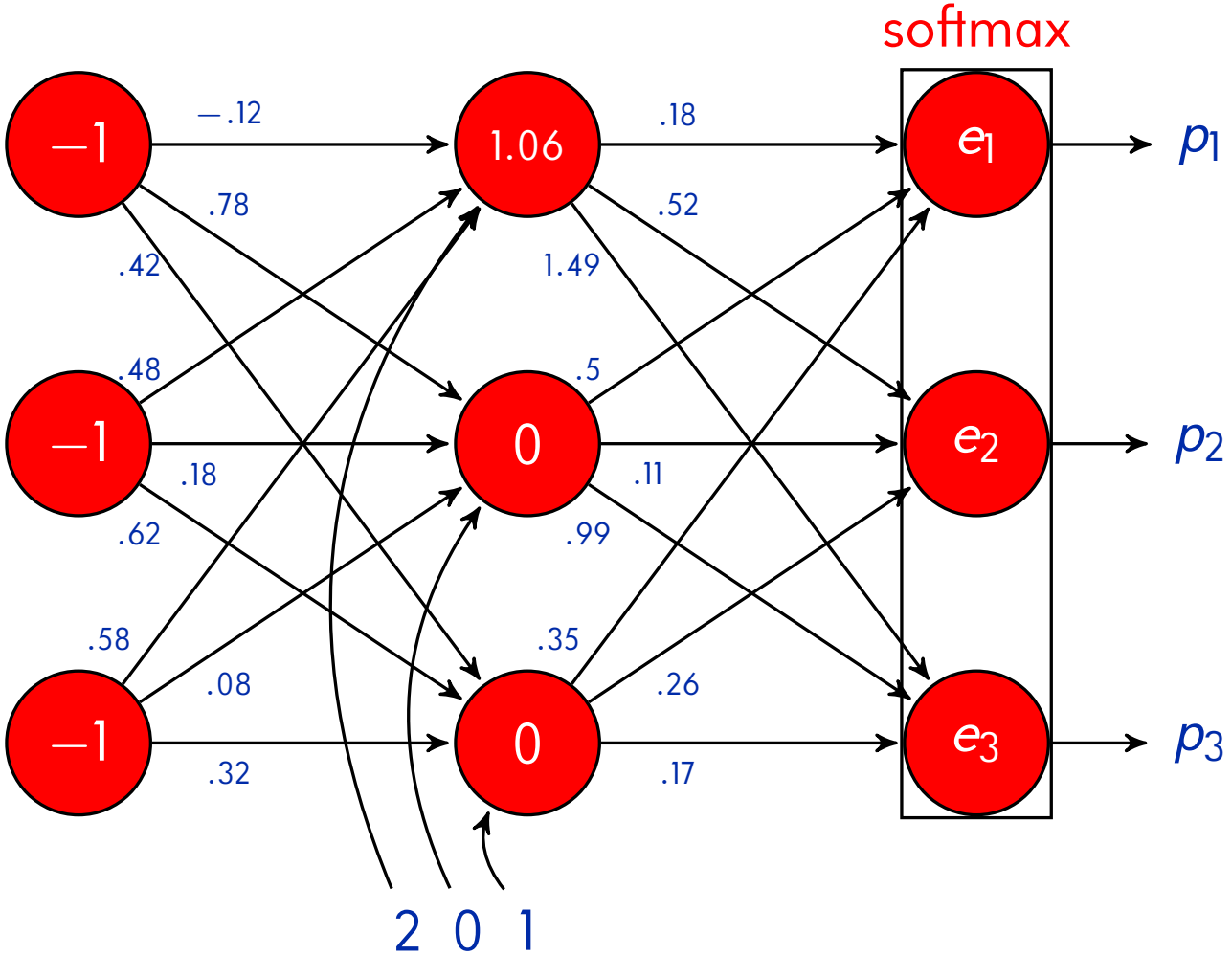
- Previous loss $-\log(0.34) \approx 1.08$



Training

Backpropagation

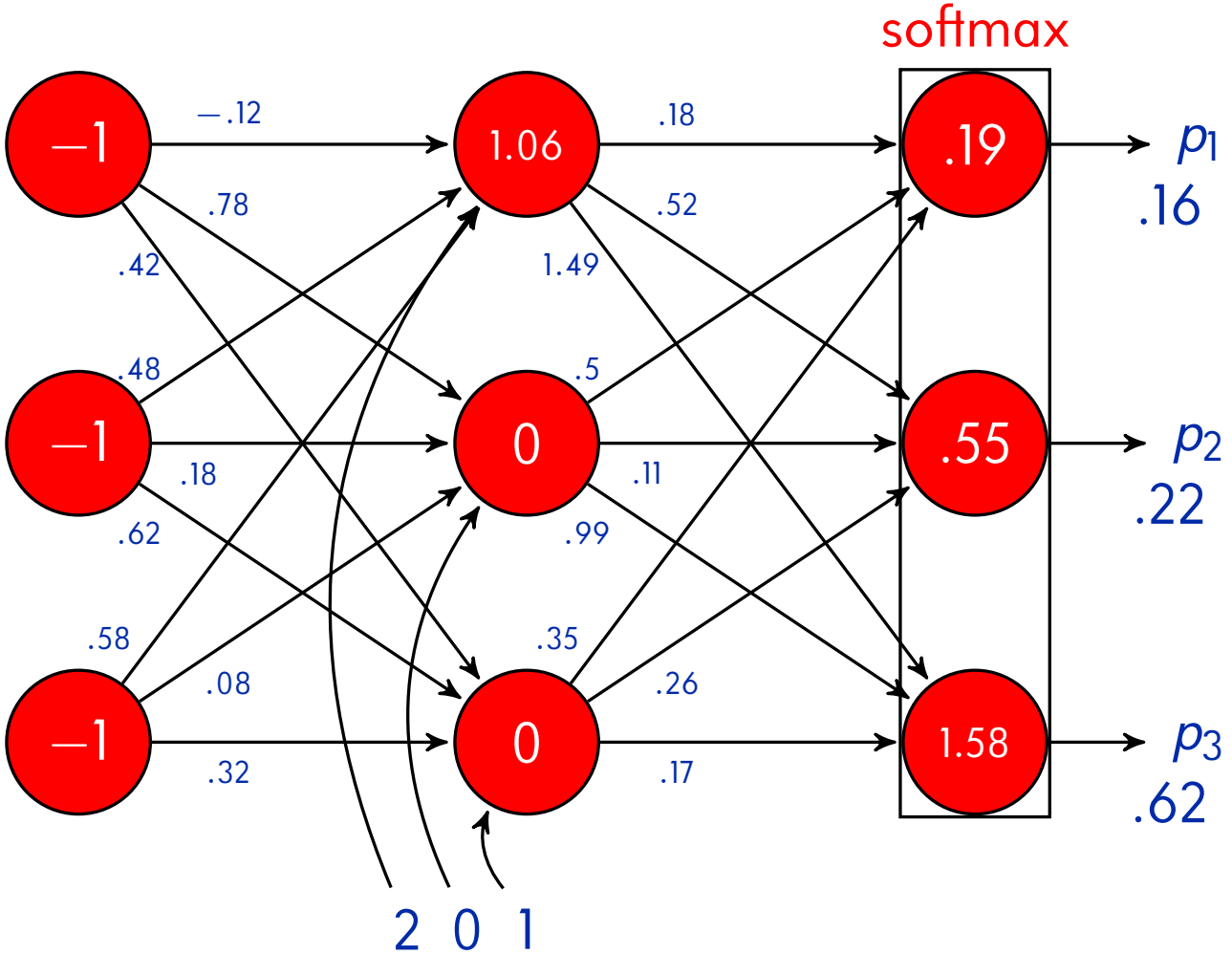
- Previous loss $-\log(0.34) \approx 1.08$



Training

Backpropagation

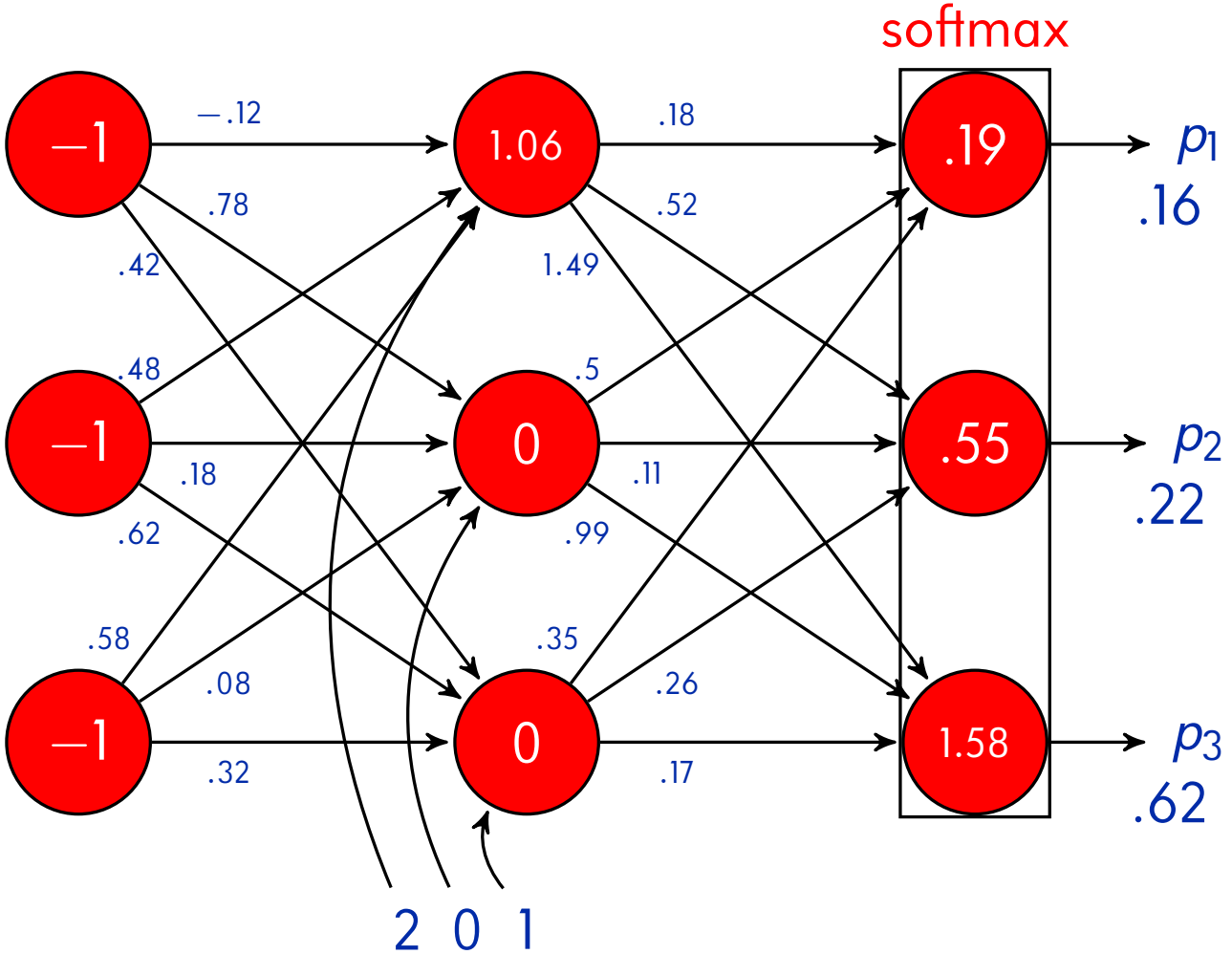
- Previous loss $-\log(0.34) \approx 1.08$



Training

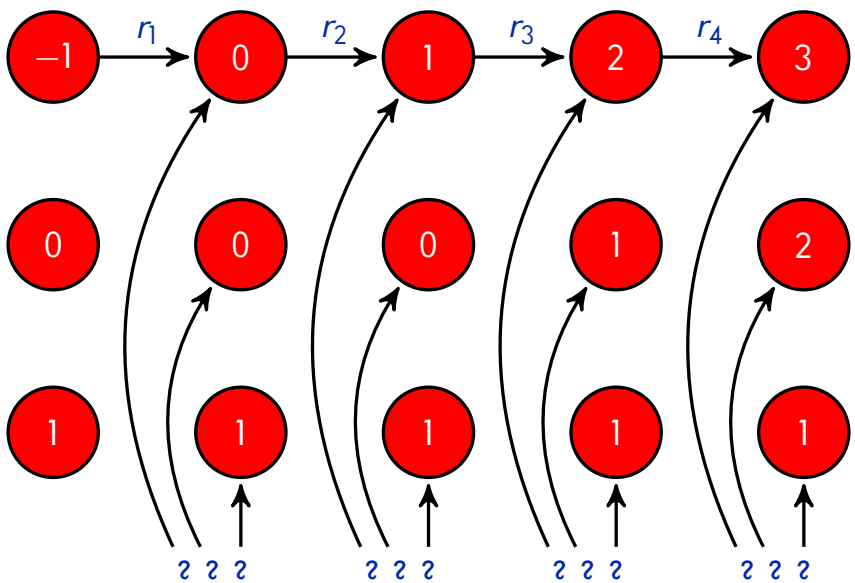
Backpropagation

- Previous loss $-\log(0.34) \approx 1.08$
- New loss $-\log(0.62) \approx 0.48$



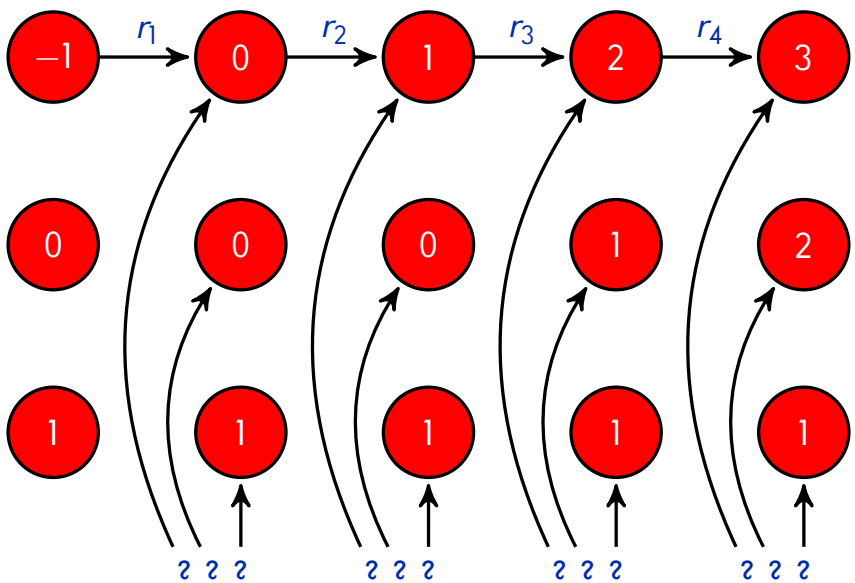
Backpropagation through time

- Train feed-forward network and obtain $\frac{\partial L}{\partial r_i}$ for all $i \in \{1, 2, 3, 4\}$



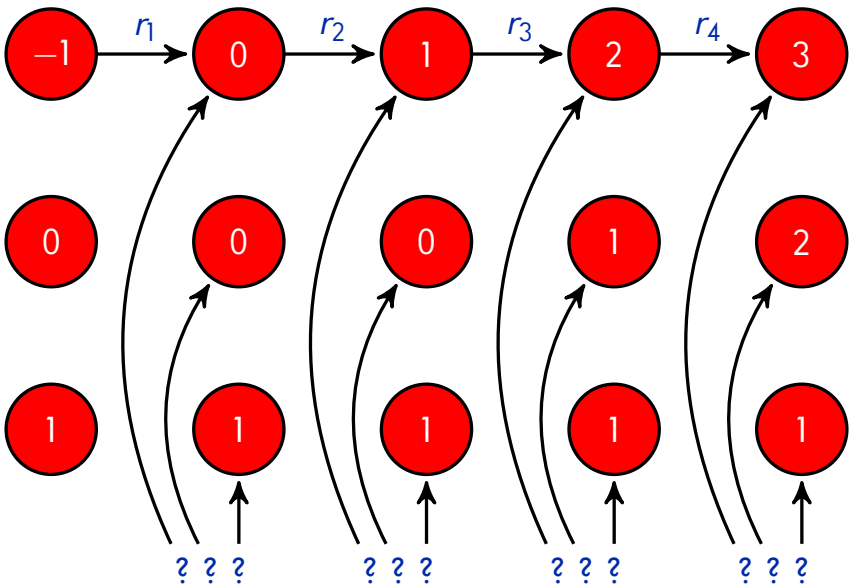
Backpropagation through time

- Train feed-forward network and obtain $\frac{\partial L}{\partial r_i}$ for all $i \in \{1, 2, 3, 4\}$
- $r_1 = r_2 = r_3 = r_4$ due to unrolling from weight r



Backpropagation through time

- Train feed-forward network and obtain $\frac{\partial L}{\partial r_i}$ for all $i \in \{1, 2, 3, 4\}$
- $r_1 = r_2 = r_3 = r_4$ due to unrolling from weight r
- Update $r' = r - \frac{\gamma}{4}\left(\sum_{i=1}^{4} \frac{\partial L}{\partial r_i}\right)$ by average of updates for copies



Training

Stochastic gradient descent

- Computing gradients for training data usually too slow & expensive
- Cut data into smaller chunks (randomize order, equal size parts)

Training

Stochastic gradient descent

- Computing gradients for training data usually too slow & expensive
- Cut data into smaller chunks (randomize order, equal size parts)
- Batch = chunk for which gradients are computed together
  (e.g. 256 training examples)

Training

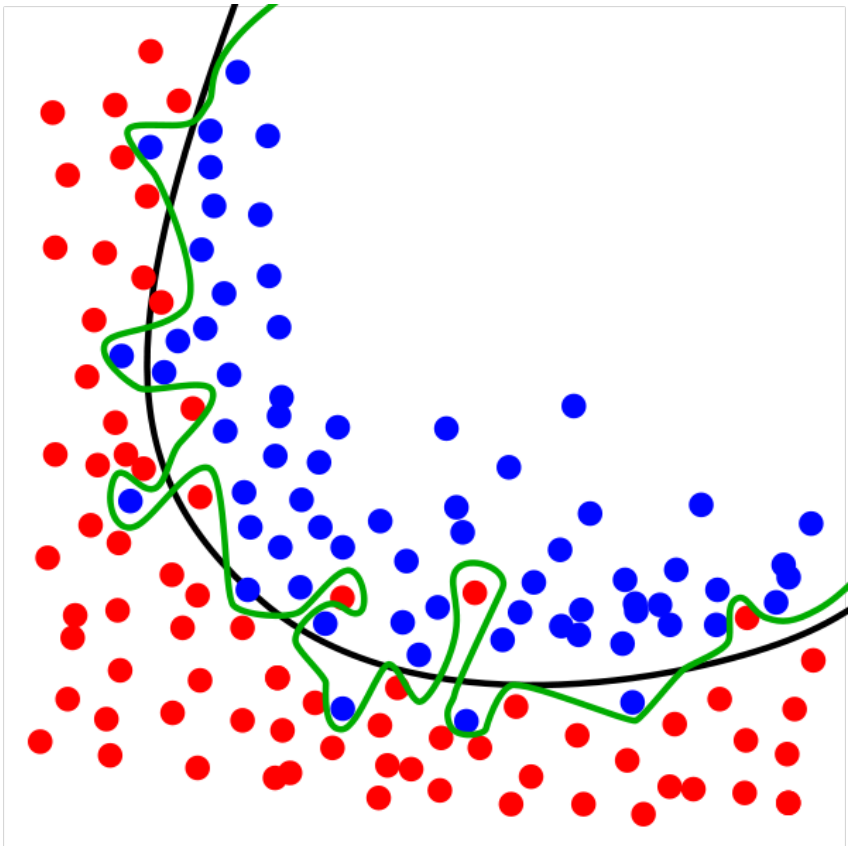
Stochastic gradient descent

- Computing gradients for training data usually too slow & expensive
- Cut data into smaller chunks (randomize order, equal size parts)
- Batch = chunk for which gradients are computed together
  (e.g. 256 training examples)
- Train for d epochs = d iterations over all training examples

Training

Final notes

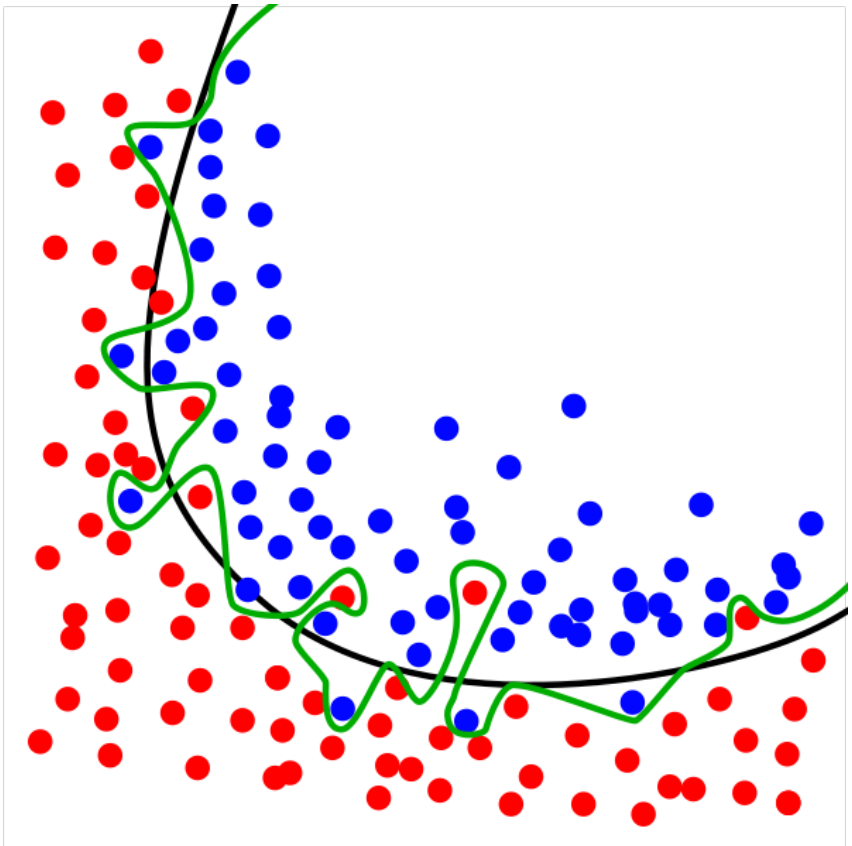
- Too few epochs $\rightarrow$ bad data fit
- Too many epochs $\rightarrow$ overfitting



© Chabacano

Final notes

- Too few epochs $\rightarrow$ bad data fit
- Too many epochs $\rightarrow$ overfitting
- Techniques to avoid overfitting: validation loss, drop-out, ...



© Chabacano

That's all folks!

Thank you for the attention

Additional lemma

Lemma A

$$\prod_{t\in\mathbb{N}} \frac{e^{k(t+1)}}{1+e^{k(t+1)}} = \frac{2}{(-1;e^{-k})_\infty}$$

for all $k \in \mathbb{N}$

Additional lemma

Lemma A

$$\prod_{t\in\mathbb{N}} \frac{e^{k(t+1)}}{1+e^{k(t+1)}} = \frac{2}{(-1;e^{-k})_\infty} \qquad \text{for all } k \in \mathbb{N}$$

Proof

$$\prod_{t\in\mathbb{N}} \frac{e^{k(t+1)}}{1+e^{k(t+1)}} = \prod_{t\in\mathbb{N}_+} \frac{e^{kt}}{1+e^{kt}} \cdot \frac{e^{-kt}}{e^{-kt}} = \prod_{t\in\mathbb{N}_+} \frac{\cancel{e^{kt}}}{1+e^{kt}} \cdot \frac{\cancel{e^{-kt}}}{e^{-kt}}$$

$$= \prod_{t\in\mathbb{N}_+} \frac{1}{1+e^{-kt}} = 2 \cdot \prod_{t\in\mathbb{N}} \frac{1}{1+e^{-kt}} = \frac{2}{(-1;e^{-k})_\infty}$$

Additional lemma

Lemma B

$$\sum_{k \in \mathbb{N}} \mathcal{R}(a^k) = 1$$

Additional lemma

Lemma B

$$\sum_{k\in\mathbb{N}} \mathcal{R}(a^k) = 1$$

Proof

$$\sum_{k\in\mathbb{N}} \mathcal{R}(a^k) = \Big(\sum_{k=0}^{\ell} \mathcal{R}(a^k)\Big) + \Big(\sum_{k=\ell+1}^{\infty} \mathcal{R}(a^k)\Big)$$

$$= \sum_{k=0}^{\ell}\Big(\frac{1}{1+\mathrm{e}^{2(k+1)}} \cdot \prod_{t=0}^{k-1} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}\Big) + \sum_{k=\ell+1}^{\infty} \frac{\mathrm{e}^{k-3\ell-5}}{1+\mathrm{e}^{k-3\ell-5}} \cdot \Big(\prod_{t=0}^{\ell} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{k-1} \frac{1}{1+\mathrm{e}^{t-3\ell-5}}\Big)$$

$$= 1 - \Big(\prod_{t=0}^{\ell} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}\Big) + \Big(\prod_{t=0}^{\ell} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}\Big) \cdot \underbrace{\sum_{k=\ell+1}^{\infty} \frac{\mathrm{e}^{k-3\ell-5}}{1+\mathrm{e}^{k-3\ell-5}} \cdot \Big(\prod_{t=\ell+1}^{k-1} \frac{1}{1+\mathrm{e}^{t-3\ell-5}}\Big)}_{1-\prod_{t=\ell+1}^{\infty} \frac{1}{1+\mathrm{e}^{t-3\ell-5}}}$$

$$= 1 - \Big(\prod_{t=0}^{\ell} \frac{\mathrm{e}^{2(t+1)}}{1+\mathrm{e}^{2(t+1)}}\Big) \cdot \Big(1 - 1 + \prod_{t=\ell+1}^{\infty} \frac{1}{1+\mathrm{e}^{t-3\ell-5}}\Big) = 1 - \Big(\prod_{t=0}^{\ell} \frac{1}{1+\mathrm{e}^{-2(t+1)}}\Big) \cdot \Big(\prod_{t=\ell+1}^{\infty} \frac{1}{1+\mathrm{e}^{t-3\ell-5}}\Big)$$

$$\geq 1 - \Big(\prod_{t=0}^{\ell} \frac{1}{1+\mathrm{e}^{-2(t+1)}}\Big) \cdot \Big(\prod_{t\in\mathbb{N}} \frac{1}{1+\mathrm{e}^{t}}\Big) = 1$$

Additional lemma

Lemma C

$$\frac{\partial L}{\partial \operatorname{wt}(n_1, e_3)} = \frac{1}{\ell} \sum_{i \in \{1,\ldots,\ell\}, z_3^{(i)} > 0} (p_3^{(i)} - t_3^{(i)}) n_1^{(i)}$$

Additional lemma

Lemma C

$$\frac{\partial L}{\partial \operatorname{wt}(n_1, e_3)} = \frac{1}{\ell} \sum_{i \in \{1,\dots,\ell\}, z_3^{(i)} > 0} (p_3^{(i)} - t_3^{(i)}) n_1^{(i)}$$

Proof

$$\frac{\partial L}{\partial \operatorname{wt}(n_1, e_3)} = \sum_{i=1}^{\ell} \sum_{k=1}^{3} \sum_{m=1}^{3} \frac{\partial L}{\partial p_k^{(i)}} \frac{\partial p_k^{(i)}}{\partial e_m^{(i)}} \frac{\partial e_m^{(i)}}{\partial z_m^{(i)}} \frac{\partial z_m^{(i)}}{\partial \operatorname{wt}(n_1, e_3)} = \sum_{\substack{i \in \{1,\dots,\ell\} \\ z_3^{(i)} > 0}} \sum_{j=1}^{3} -\frac{t_k^{(i)}}{\ell p_k^{(i)}} \frac{\partial p_k^{(i)}}{\partial e_3^{(i)}} n_1^{(i)}$$

$$= \sum_{\substack{i \in \{1,\dots,\ell\} \\ z_3^{(i)} > 0}} \left(-\frac{t_3^{(i)}}{\ell \cancel{p_3^{(i)}}} \cancel{p_3^{(i)}} (1 - p_3^{(i)}) - \sum_{k=1}^{2} \frac{t_k^{(i)}}{\ell \cancel{p_k^{(i)}}} (-\cancel{p_k^{(i)}} p_3^{(i)}) \right) n_1^{(i)}$$

$$= \frac{1}{\ell} \sum_{i \in \{1,\dots,\ell\}, z_3^{(i)} > 0} \left(-t_3^{(i)} (1 - p_3^{(i)}) + \sum_{k=1}^{2} t_k^{(i)} p_3^{(i)} \right) n_1^{(i)}$$

$$= \frac{1}{\ell} \sum_{i \in \{1,\dots,\ell\}, z_3^{(i)} > 0} \left(-t_3^{(i)} + p_3^{(i)} \sum_{k=1}^{3} t_k^{(i)} \right) n_1^{(i)} = \frac{1}{\ell} \sum_{i \in \{1,\dots,\ell\}, z_3^{(i)} > 0} (p_3^{(i)} - t_3^{(i)}) n_1^{(i)}$$