# Recurrent Neural Networks

Lecture 21

IE 643

October 20, 2024

## Recurrent Neural Networks

## Sequential outputs - Motivating Applications

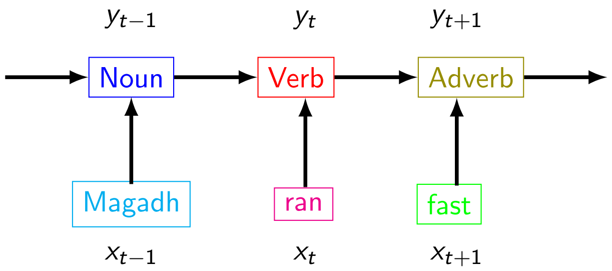

Figure: Part-of-Speech Tagging

## Sequential outputs - Motivating Applications

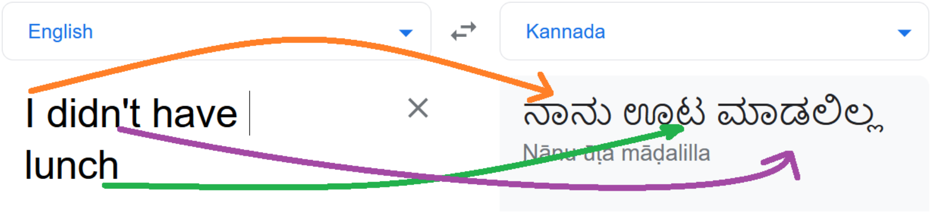
English
Kannada
I didn't have lunch
ನಾನು ಊಟ ಮಾಡಲಿಲ್ಲ
Nānu ūṭa māḍalilla

Figure: Language Translation

## Sequential outputs - Motivating Applications

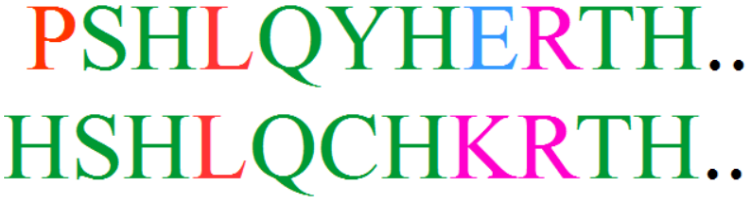

Figure: Protein Sequence Alignment

## Earliest Recurrent Network Architectures

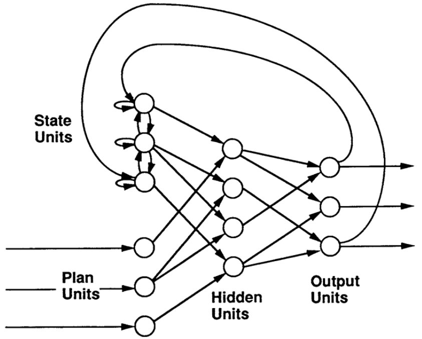

**FIGURE 1.** The processing units and basic interconnection scheme (not all connections are shown). The plan and state units together constitute the input units for the network.

Figure: Jordan Network[†]

[†] Michael Jordan. *Serial order: A parallel distributed processing approach (Tech. Rep. No. 8604).*, Institute for Cognitive Science, UCSD, San Diego, 1986

## Earliest Recurrent Network Architectures

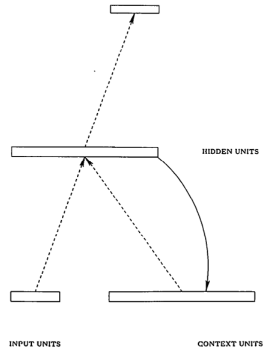

**Figure** . A simple recurrent network in which activations are copied from hidden layer to context layer on a one-for-one basis, with fixed weight of 1.0. Dotted lines represent trainable connections.

Figure: Elman Network[‡]

[‡] Jeffrey L. Elman, *Finding structure in time*, Cognitive Science, Vol 14(2), pp. 179-211, 1990.

## Recurrent Neural Network

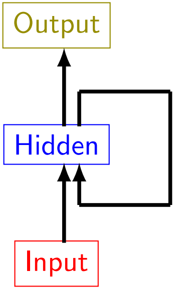

Figure: A simple recurrent network

## Recurrent Neural Network

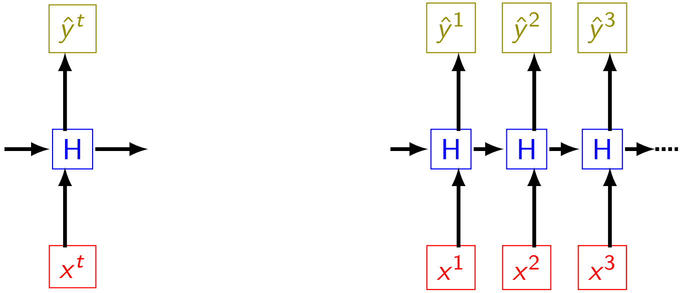

Figure: A simple recurrent network

## Recurrent Layers

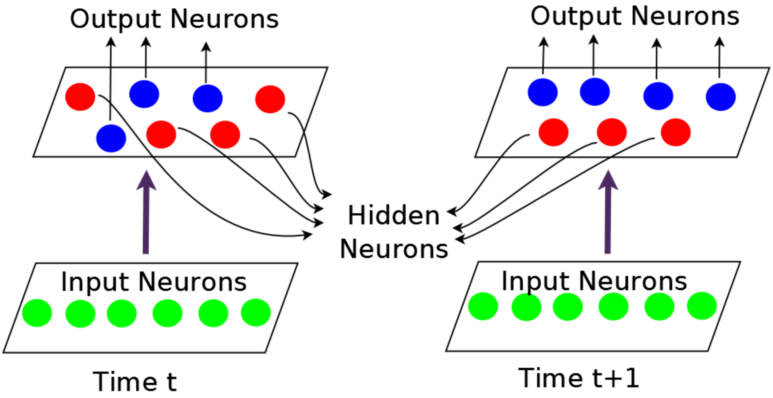

Figure: Recurrent layers

## Recurrent Layers - Formalism

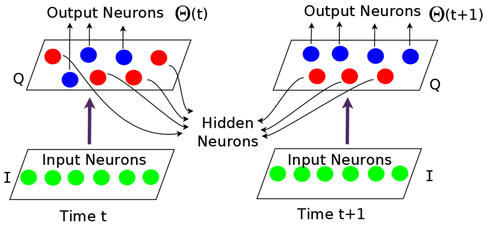

Figure: Recurrent layers

- $S$ be the set of all neurons
- $S = \mathtt{I} \cup Q$
  - $\mathtt{I}$ is the set of input neurons
  - $Q$ is the set of hidden neurons
- Some neurons in set $Q$ can be used as output neurons at particular time steps (this set is denoted by $\Theta(t) \subseteq Q$).

## Recurrent Layers

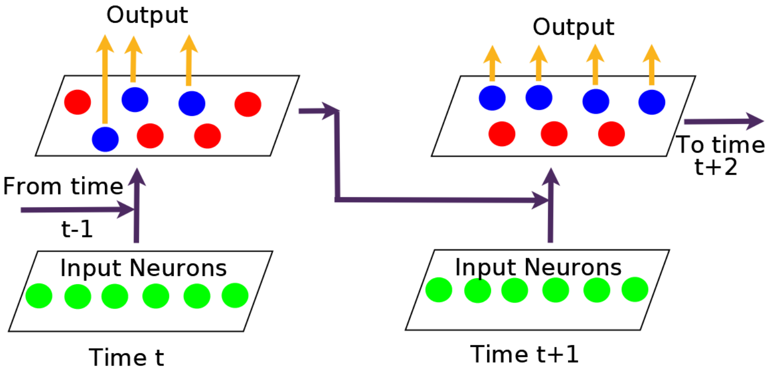

Figure: Recurrent layers

## Recurrent Layers - Formalism

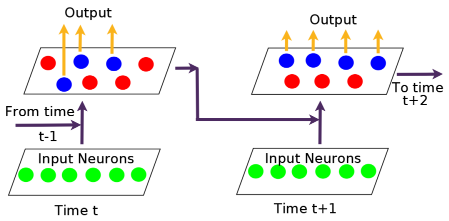

Figure: Recurrent layers

- At the time step $t$, we define:

$$\xi_k(t) = \begin{cases} a_k(t-1) & if \;\; k \in Q \\ x_k(t) & if \;\; k \in I \end{cases}$$

- $a_k(t-1)$ denotes activation of neuron $k$ at time $t-1$.
- $x_k(t)$ denotes input at neuron $k$ at time $t$.
- Note: $\hat{y}_k(t-1) = a_k(t-1)$.

## Recurrent Layers - Formalism[†]

- At the time step $t$, we define:

$$\xi_k(t) = \begin{cases} a_k(t-1) & \textit{if} \;\; k \in Q \\ x_k(t) & \textit{if} \;\; k \in I \end{cases}$$

- For all neurons belonging to $S$ we can now define:

$$z_k(t) = \sum_{\ell \in S} w_{k\ell} \xi_\ell(t) \;\; \forall k \in S$$
$$a_k(t) = \phi[z_k(t)]$$

- $\phi$ denotes activation function. (*e.g.* sigmoidal, ReLU)

[†] R. J. Williams and D. Zipser, *Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity*, Backpropagation, L. Erlbaum Associates Inc. Hillsdale, NJ, USA. 1995.

## Recurrent Layers - Formalism

- For time step $t$, denote the set of output neurons as $\Theta(t) \subseteq Q$.
- Let $y_k(t)$ be the actual output associated with neuron $k$ at time $t$.
- Error at time step $t$ for a particular neuron $k$:

$$e_k(t) = \begin{cases} y_k(t) - a_k(t) & \text{when} \quad k \in \Theta(t), \\ 0 & \text{else.} \end{cases}$$

## Recurrent Layers - Formalism

- For time step $t$, denote the set of output neurons as $\Theta(t) \subseteq Q$.
- Let $y_k(t)$ be the actual output associated with neuron $k$ at time $t$.
- Error at time step $t$ for a particular neuron $k$:

$$e_k(t) = \begin{cases} y_k(t) - a_k(t) & \text{when} \quad k \in \Theta(t), \\ 0 & \text{else.} \end{cases}$$

- Error at time step $t$ can be defined as a squared error function:

$$E(t) = \sum_{k \in S} e_k^2(t)$$

## Recurrent Layers - Formalism

- For time step $t$, denote the set of output neurons as $\Theta(t) \subseteq Q$.
- Let $y_k(t)$ be the actual output associated with neuron $k$ at time $t$.
- Error at time step $t$ for a particular neuron $k$:

$$e_k(t) = \begin{cases} y_k(t) - a_k(t) & \text{when} \quad k \in \Theta(t), \\ 0 & \text{else.} \end{cases}$$

- Error at time step $t$ can be defined as a squared error function:

$$E(t) = \sum_{k \in S} e_k^2(t)$$

- Net error over all time steps (assuming that the network gets initialized at time $t_0$ and runs during time interval $(t_0, t_f])$ can be defined as:

$$E = \sum_{t=t_0+1}^{t_f} E(t) = \sum_{t=t_0+1}^{t_f} \sum_{k \in S} e_k^2(t)$$

## Recurrent Neural Networks - Loss Minimization

### Optimization problem

$$\min_{w} E = \sum_{t=t_0+1}^{t_f} E(t) = \sum_{t=t_0+1}^{t_f} \sum_{k \in S} e_k^2(t)$$

## Recurrent Neural Networks - Loss Minimization

### Optimization problem

$$\min_{w} E = \sum_{t=t_0+1}^{t_f} E(t) = \sum_{t=t_0+1}^{t_f} \sum_{k \in S} e_k^2(t)$$

### Equivalent Optimization problem

$$\min_{w} E = \sum_{t=t_0+1}^{t_f} \sum_{k \in \Theta(t)} (y_k(t) - a_k(t))^2$$

Recall:

- $a_k(t) = \phi[z_k(t)]$
- $z_k(t) = \sum_{\ell \in S} w_{k\ell} \xi_\ell(t) \quad \forall k \in S$

## Recurrent Neural Networks - Loss Minimization

### Equivalent Optimization problem

$$\min_{w} E = \sum_{t=t_0+1}^{t_f} \sum_{k \in \Theta(t)} \left( y_k(t) - \phi\left[ \sum_{\ell \in S} w_{k\ell} \xi_\ell(t) \right] \right)^2$$

## Recurrent Neural Networks - Parameter Update

### Gradient Descent Type Update

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

## Recurrent Neural Networks - Parameter Update

### Gradient Descent Type Update

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

**Homework:** Write expressions for computing the error gradients with respect to the weights. Discuss the recurrence relations obtained as part of your derivations.