Recurrent neural networks

Lecture 10

Changho Suh

January 25, 2024

Recurrent neural networks and motivation

Recap: DNNs

Work well with **enough data**.

Otherwise, we may face the overfitting problem.

This motivates simplifying DNNs, being tailored for tasks of interest.

Recap: CNNs

A model specialized for image data

Two key building blocks:

1. **Conv** layer (*mimicking* neurons in *visual cortex*)
2. **Pooling** layer (*mainly for reducing complexity*)

Design principles: As a network gets deeper:

1. Feature map size gets smaller;
2. # of feature maps gets bigger.

Recap: Tensorflow coding

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense
(X_train, y_train),(X_test, y_test) = mnist.load_data()
X_train, X_test = X_train/255.0, X_test/255.0
model_lenet = Sequential()
```

#1st layer: Stack 1 ([Conv]+[ReLU]+[Pool])

```
model_lenet.add(Conv2D(input_shape=(28,28,1),kernel_size=(5,5), strides=(1,1), filters=32,
padding='same', activation='relu'))
model_lenet.add(MaxPool2D(pool_size=(2,2),strides=(2,2),padding='valid'))
```

#2nd layer: Stack 2 ([Conv]+[ReLU]+[Pool])

```
model_lenet.add(Conv2D(kernel_size=(5,5), strides=(1,1), filters=48,
padding='same', activation='relu'))
model_lenet.add(MaxPool2D(pool_size=(2,2),strides=(2,2),padding='valid'))
```

#3rd layer (Fully-connected)

```
model_lenet.add(Flatten())
model_lenet.add(Dense(256,activation='relu'))
```

#4th layer (Fully-connected)

```
model_lenet.add(Dense(84,activation='relu'))
```

#5th layer (output layer)

```
model_lenet.add(Dense(10,activation='softmax'))
```

Applications of CNNs

Image recognition

Object detection

Defect detection

Medical diagnosis
(e.g., cancer detection)

Image inpainting

Coloring

Style transfer

Super-resolution image synthesis

Any **decision** or **manipulation** w.r.t. image data

Limitations

Not well applicable to time series data.

This is where recurrent neural networks (RNNs) kick in.

Outline of today's lectures

1. Talk about RNN's applications and history.

2. Study two key building blocks of RNNs:

   Recurrent neurons

   A **memory cell**

3. Investigate basic RNNs.

4. Study LSTM (Long Short-Term Memory) cells.

Focus of Lecture 10

1. Talk about RNN's applications and history.

2. Study two key building blocks of RNNs.

   Recurrent neurons

   A **memory cell**

3. Investigate basic RNNs.

4. Study LSTM (Long Short-Term Memory) cells.

Applications

t h □ ⇨ t h e

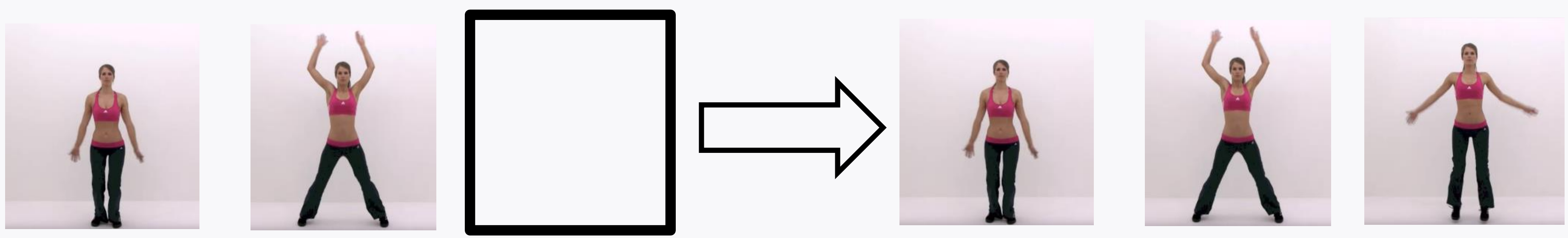
삼성밤도체 ⇨ 삼성반도체

Don’t worry ⇨ (감정) +감정

Applications

(한국어)

I like machine learning ⇨ 나는 머신러닝을 좋아해



A common feature in such applications

Memory!

DNNs & CNNs: Layers *do not preserve states*.

Hence: They do not capture such memory-feature.

This motivated the invention of RNNs.

Birth of RNNs

Pondered on the *thought process*:
Series of many thoughts & logics

Led him to conjecture existence of neurons preserving memory

→ Invented the first RNN.

William Little 1974

The first RNN was popularized by John Hopfield, hence called:

The Hopfield network

John Hopfield 1982

Another RNN in 1986 (Nature)

David Rumelhart

Geoffrey Hinton

Ronald Williams

Developed another RNN which looks very similar to nowadays RNNs.

Two building blocks of RNNs

1. Recurrent neurons

Role: Mimick conjectured neurons' behavior: having *a loop*.

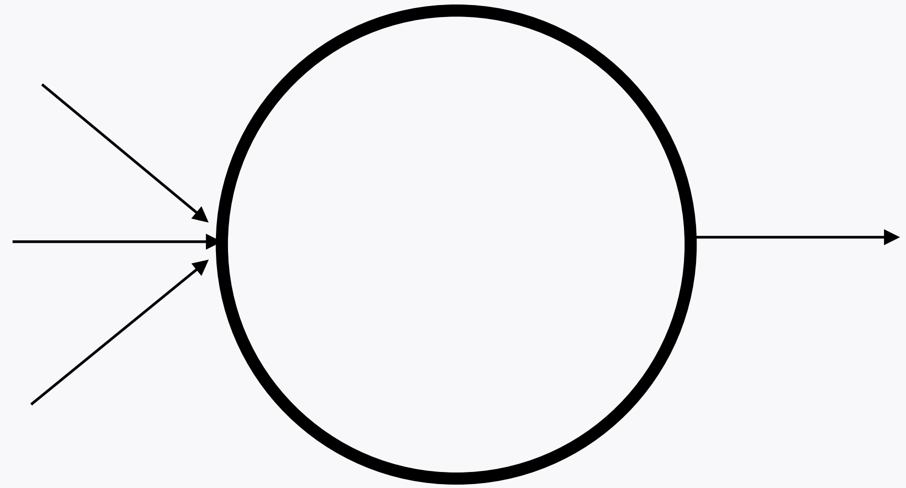
2. A memory cell

Role: *Preserve some state (memory)*.

Revisit: A conventional neuron

vector

$\mathbf{x}_t$



scalar

y_t

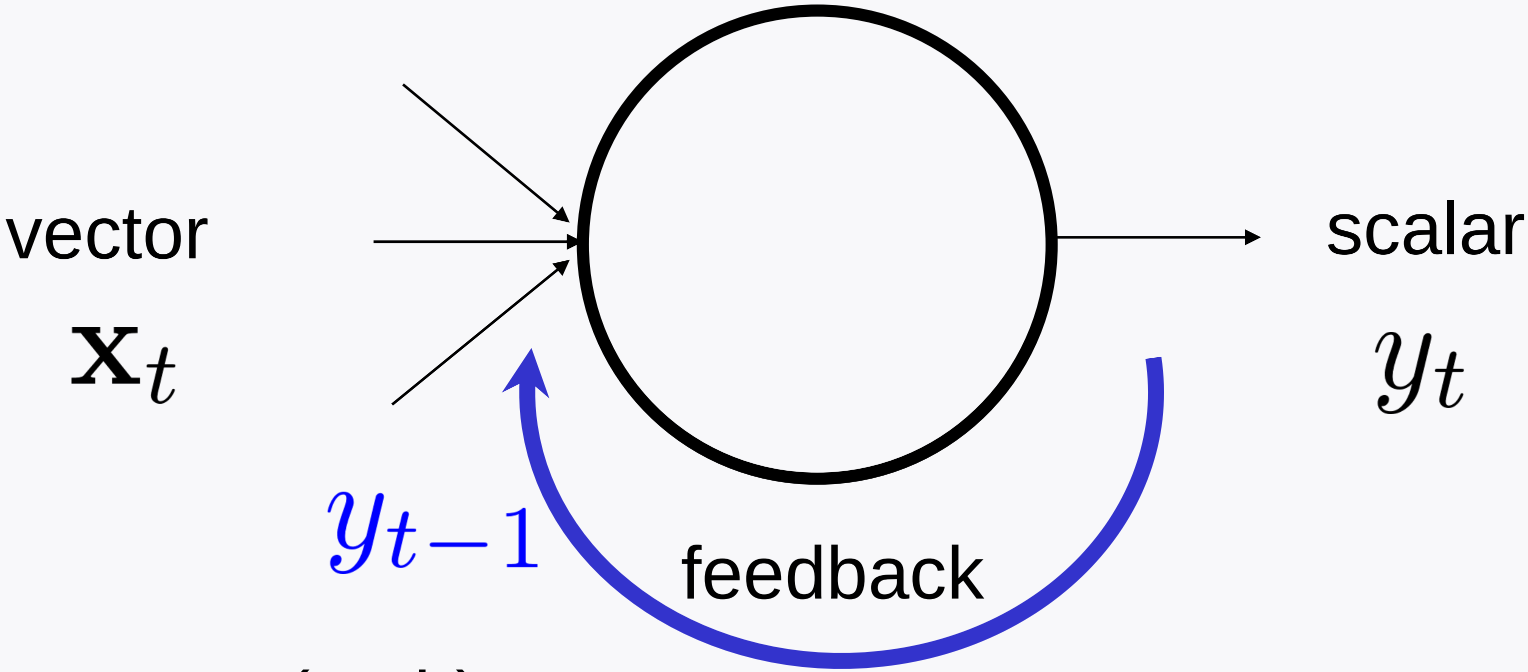
$$y_t = \phi\left(\mathbf{w}^T \mathbf{x}_t + b\right)$$

activation

A *recurrent* neuron

A conventional neuron:

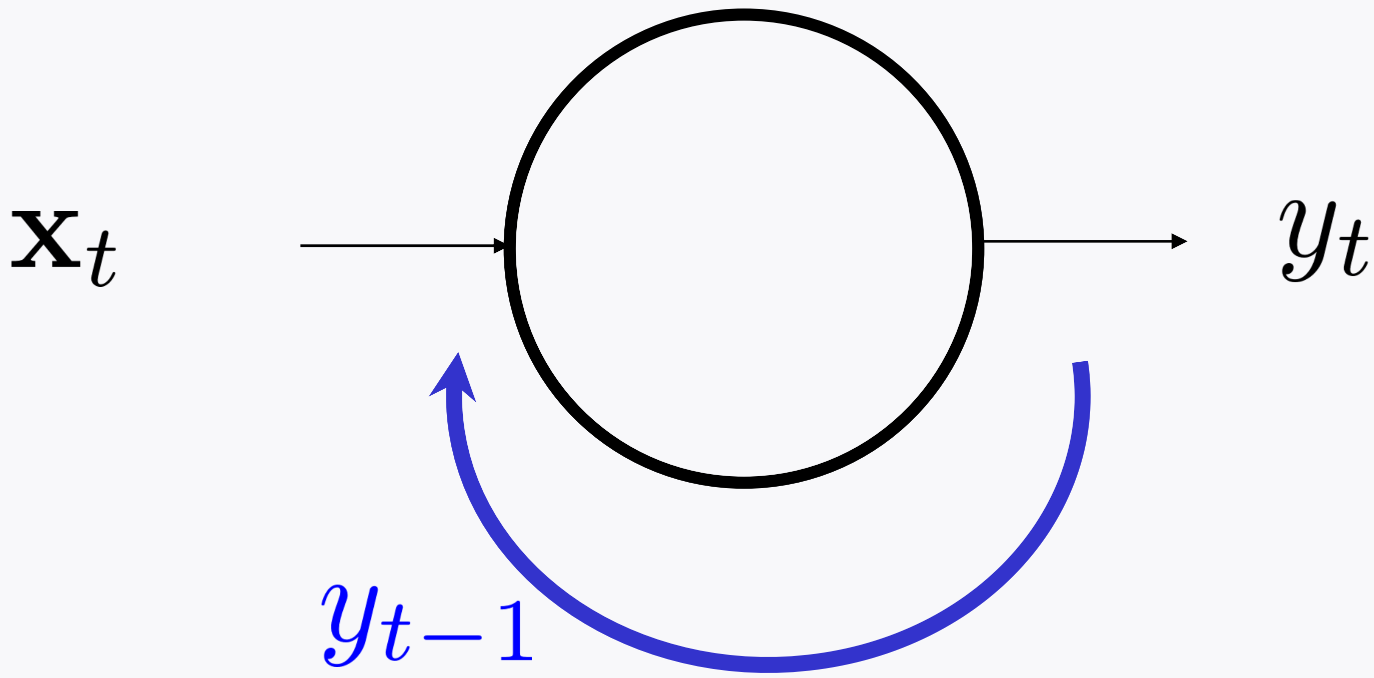
$$y_t = \phi\left(\mathbf{w}^T \mathbf{x}_t + b\right)$$



(tanh)

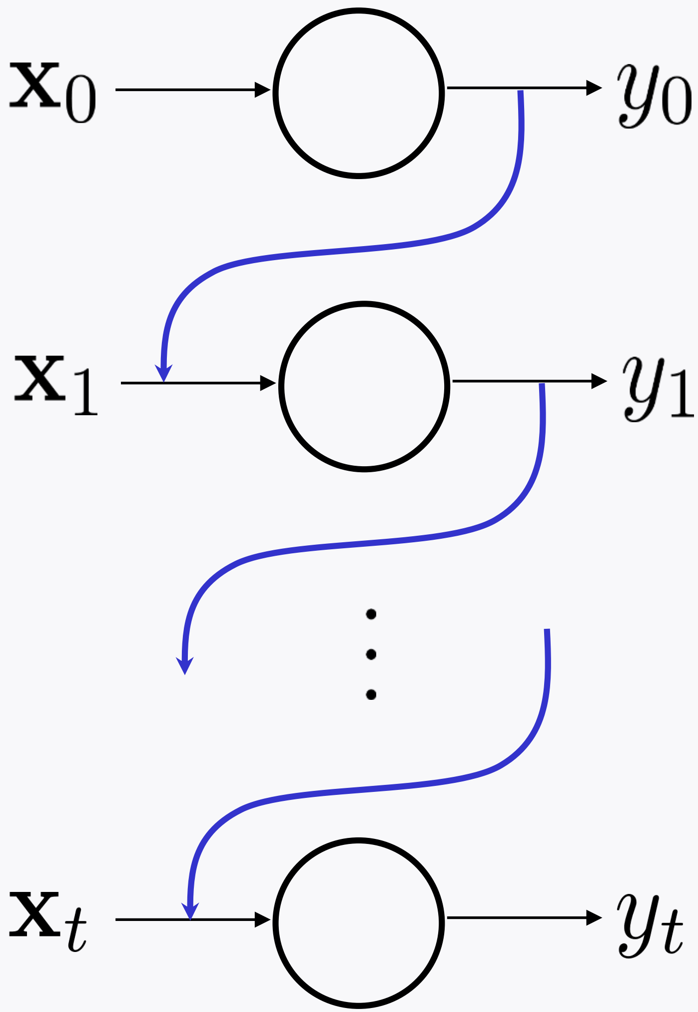
$$y_t = \phi\left(\mathbf{w}_x^T \mathbf{x}_t + w_y y_{t-1} + b\right)$$

Simplified description

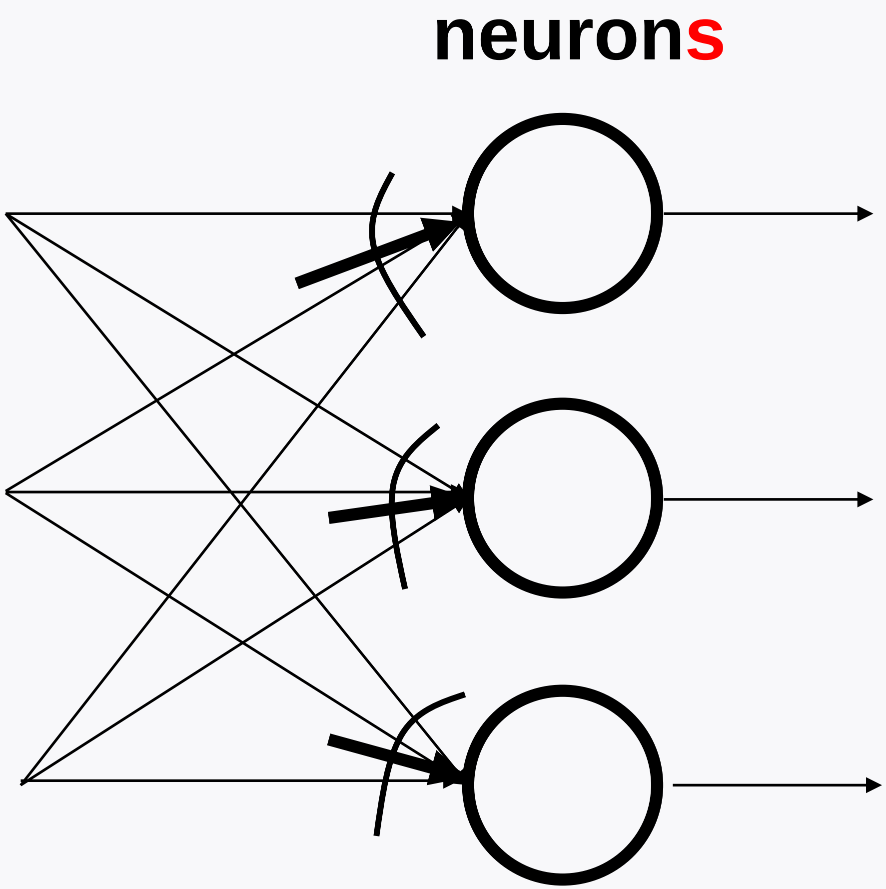


Let **a single arrow** represent the **vector signal flow**!

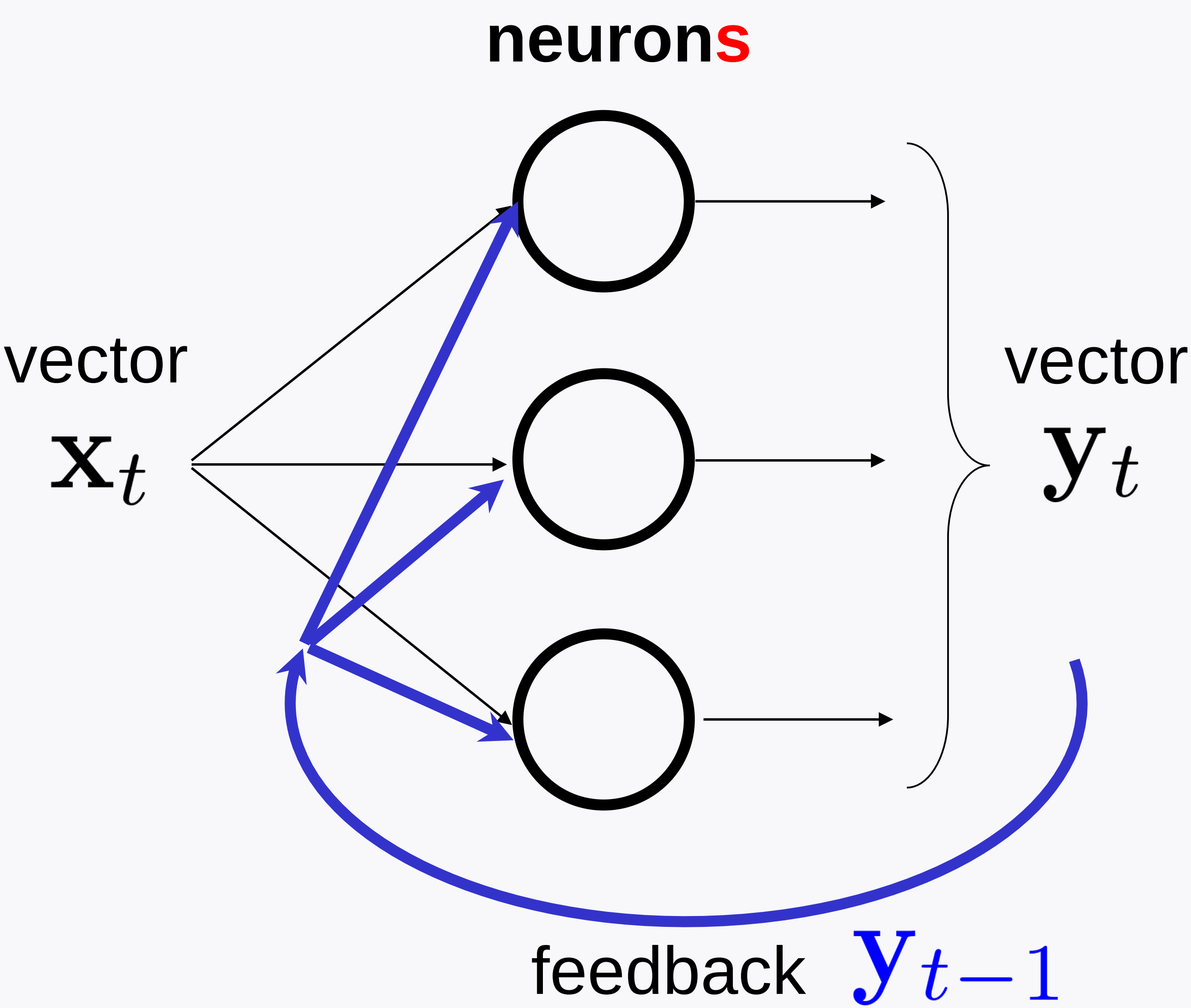
A recurrent neuron: *Unrolled* version



Recurrent neurons

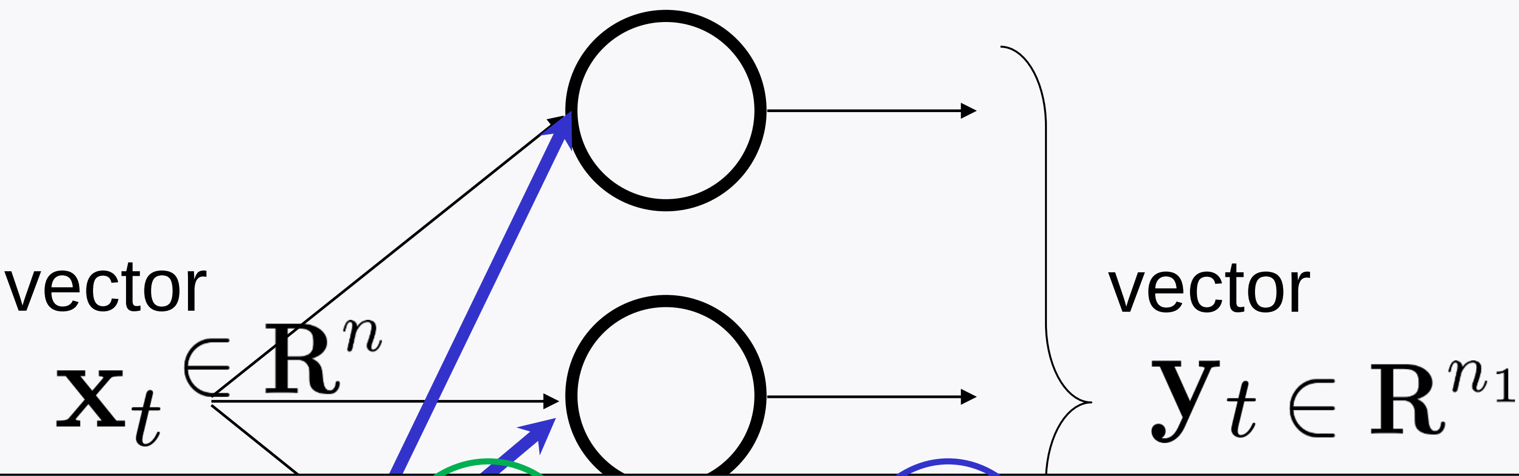


Recurrent neurons



Recurrent neurons

neurons

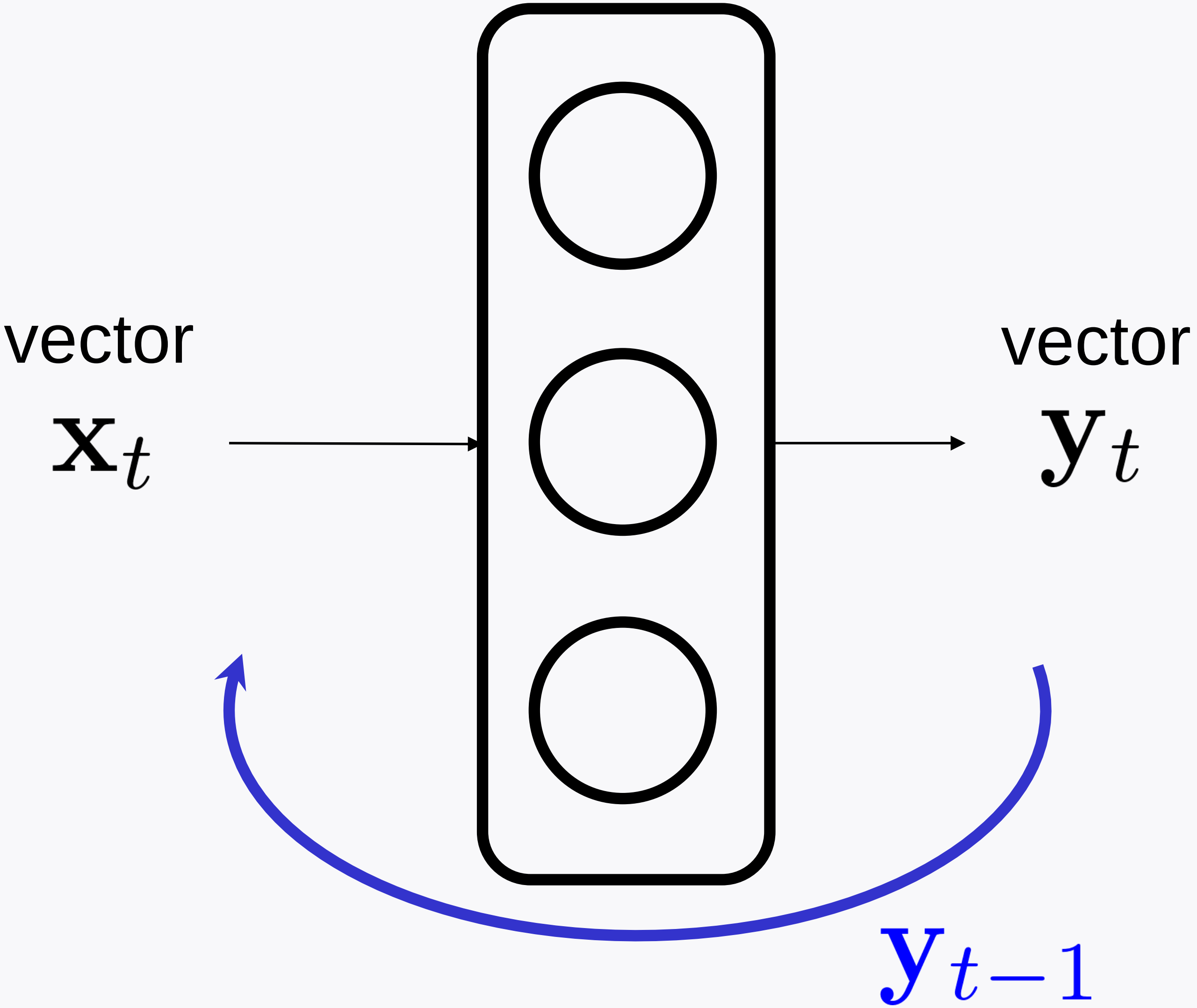


$$\mathbf{y}_t = \phi\left(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_y \mathbf{y}_{t-1} + \mathbf{b}\right)$$

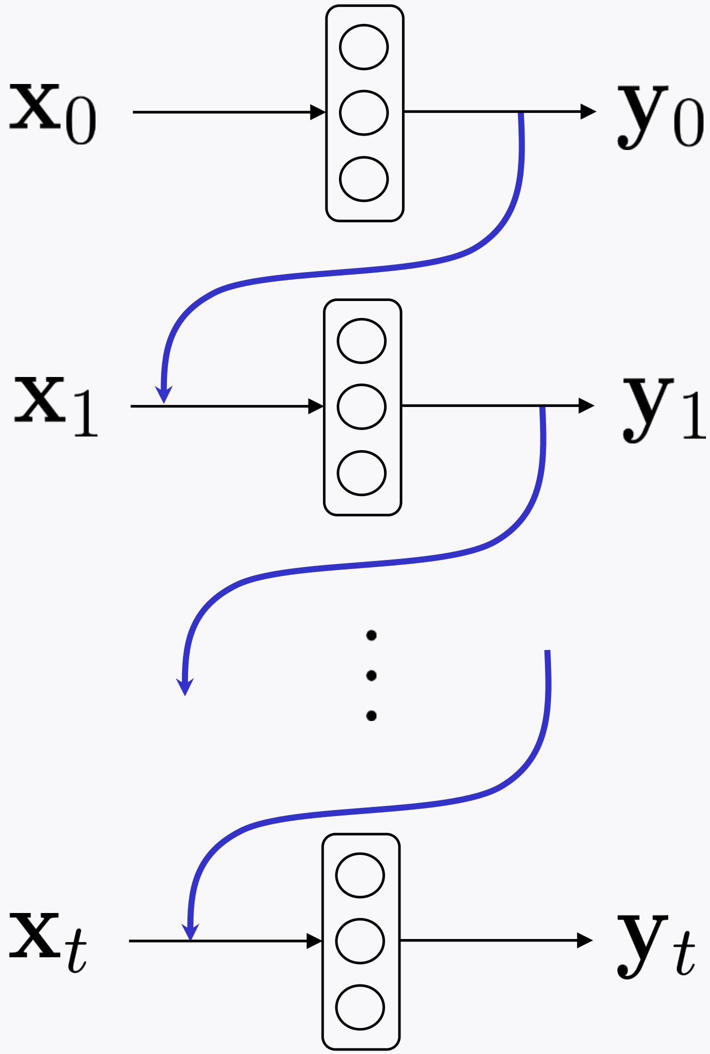
$\mathbf{W}_x \in \mathbf{R}^{n_1 \times n}$ $\qquad$ $\mathbf{W}_y \in \mathbf{R}^{n_1 \times n_1}$

feedback $\mathbf{y}_{t-1}$

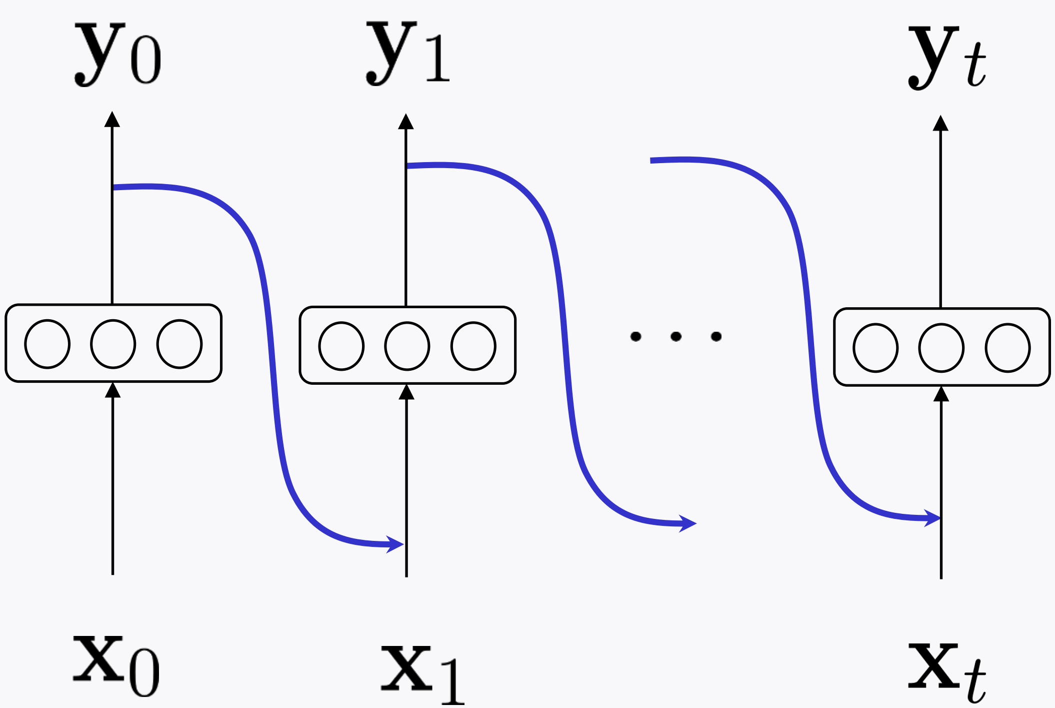
Recurrent neurons: *Simplified* description



Recurrent neurons: *Unrolled* version



Another representation



A memory cell

An entity that preserves some state $\mathbf{h}_t$ (memory).

Simply called a cell.

A basic cell: A cell such that state = output

$$\mathbf{h}_t = \mathbf{y}_t$$

Basic RNNs: RNNs with basic cells.

Look ahead

Next lecture: Will explore details on basic RNNs.