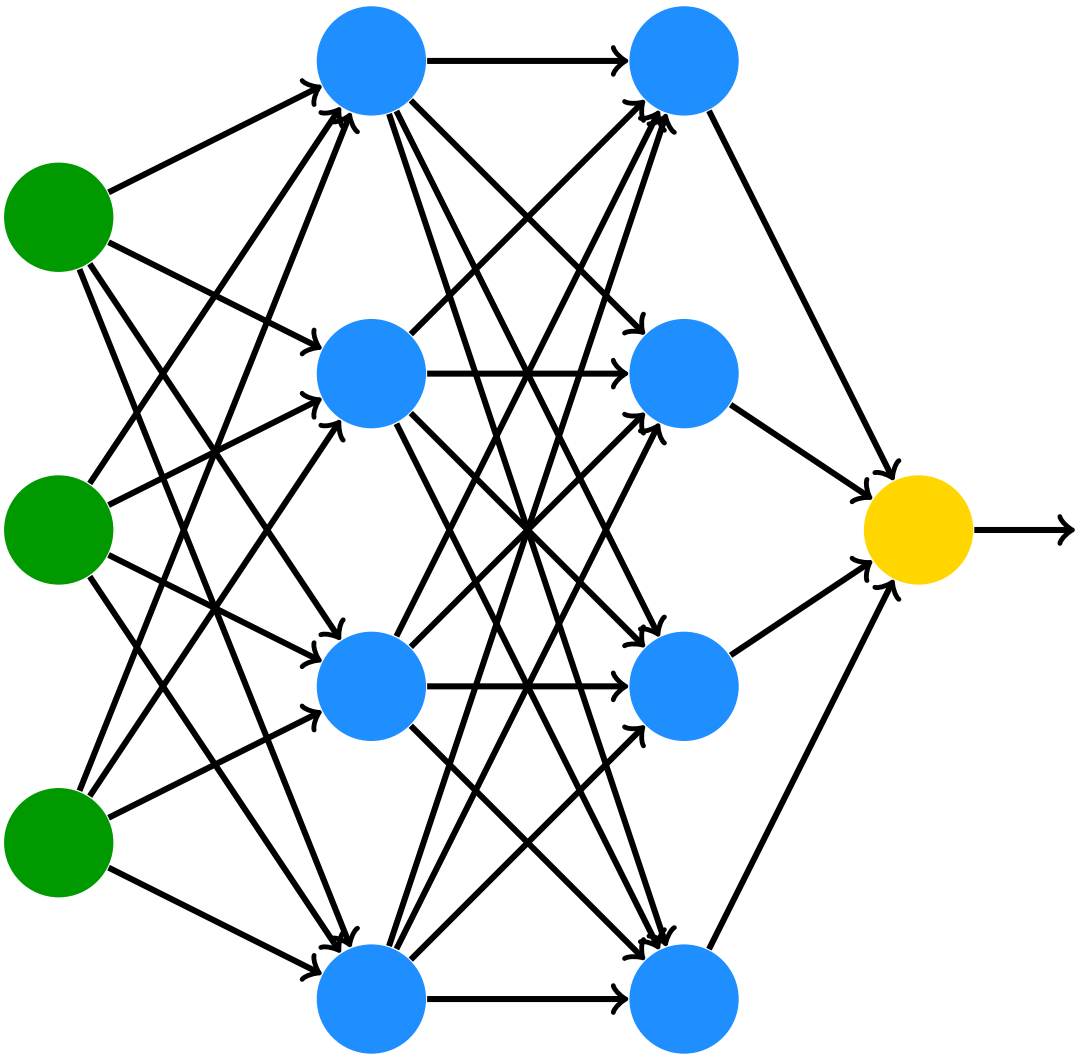
Recurrent neural networks and the classification of supernovae

Tom Charnock

Content

Deep learning

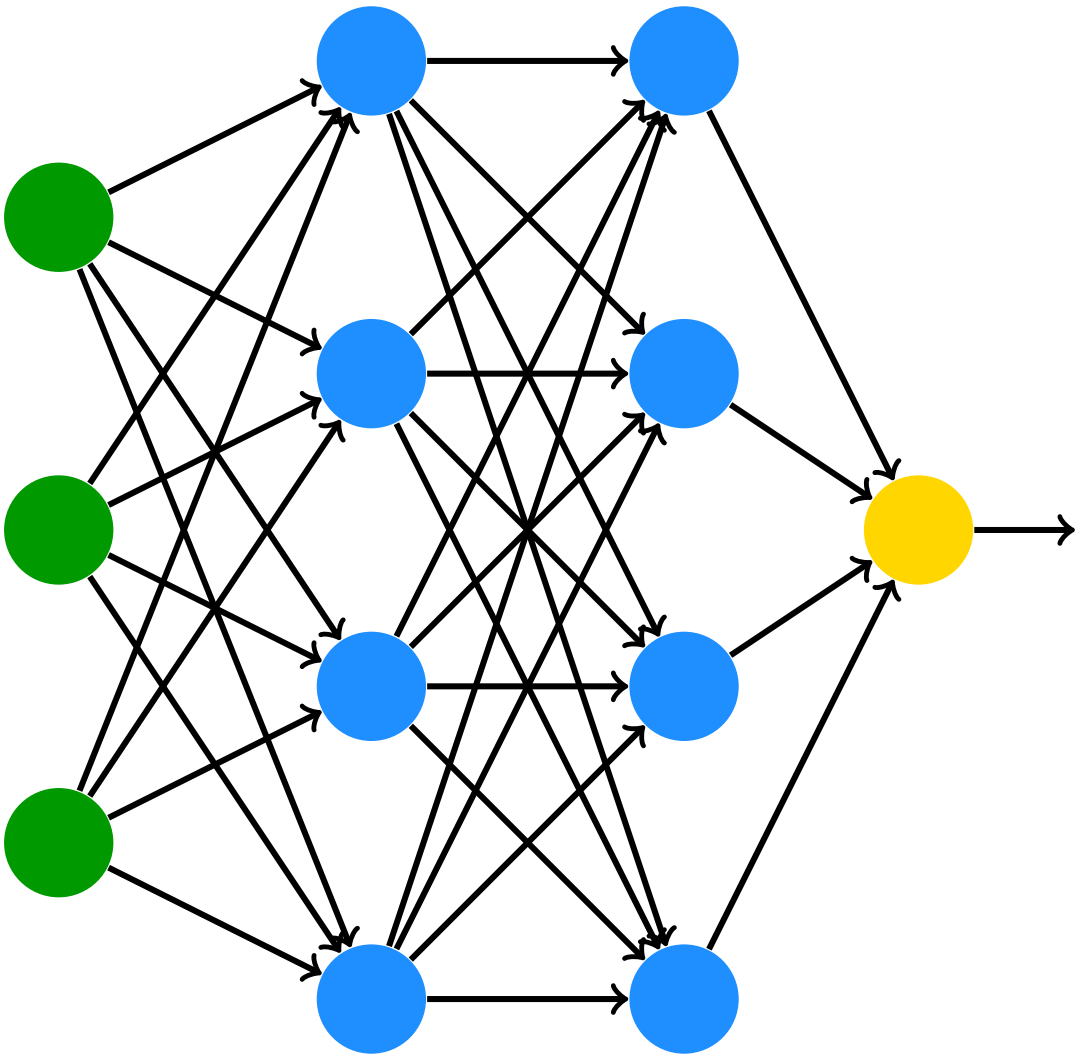


Machine learning

Specific algorithm

No prior knowledge

Deep learning



Machine learning

Specific algorithm

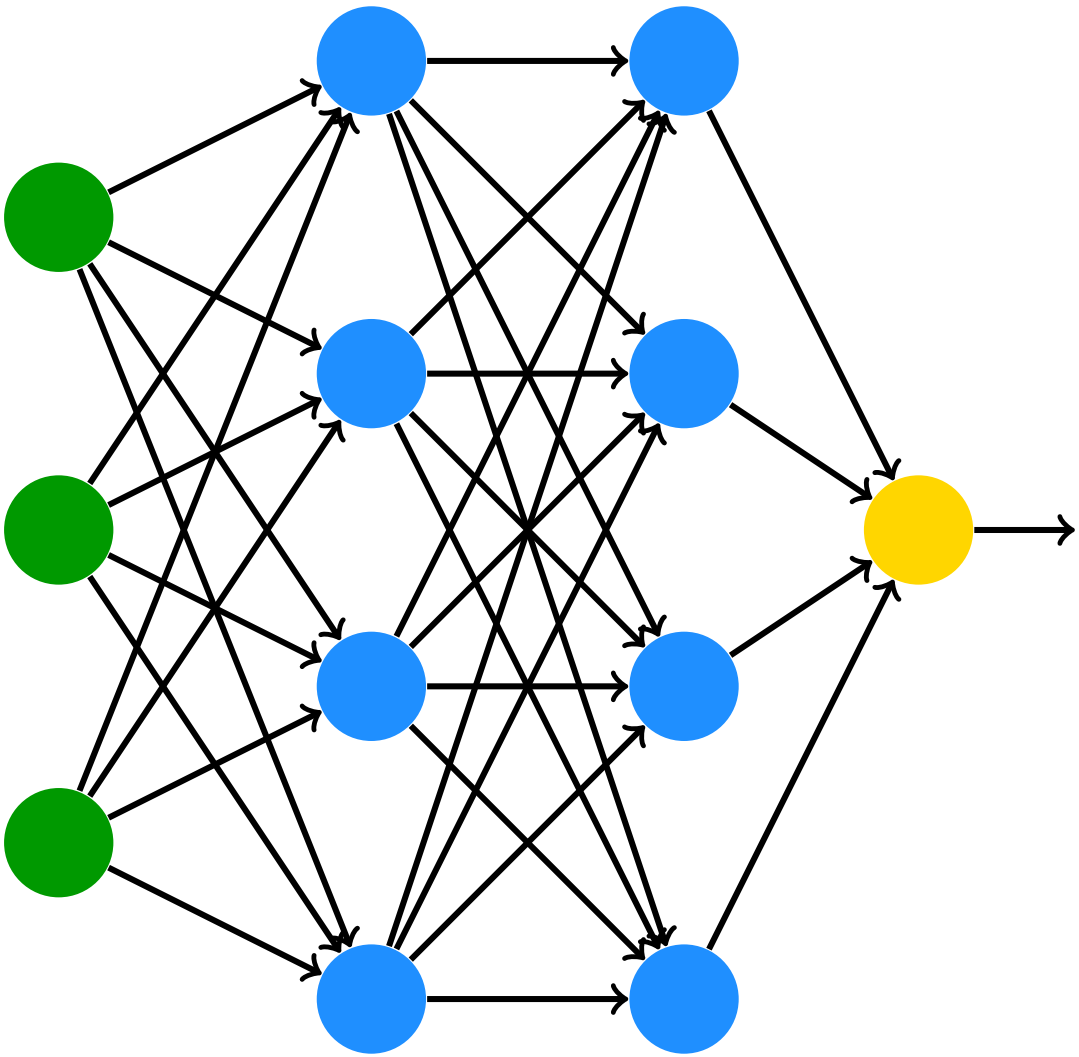
No prior knowledge

Deep learning

Multiple layers of abstraction

Blackbox

Deep learning



Machine learning

Specific algorithm

No prior knowledge

Deep learning

Multiple layers of abstraction

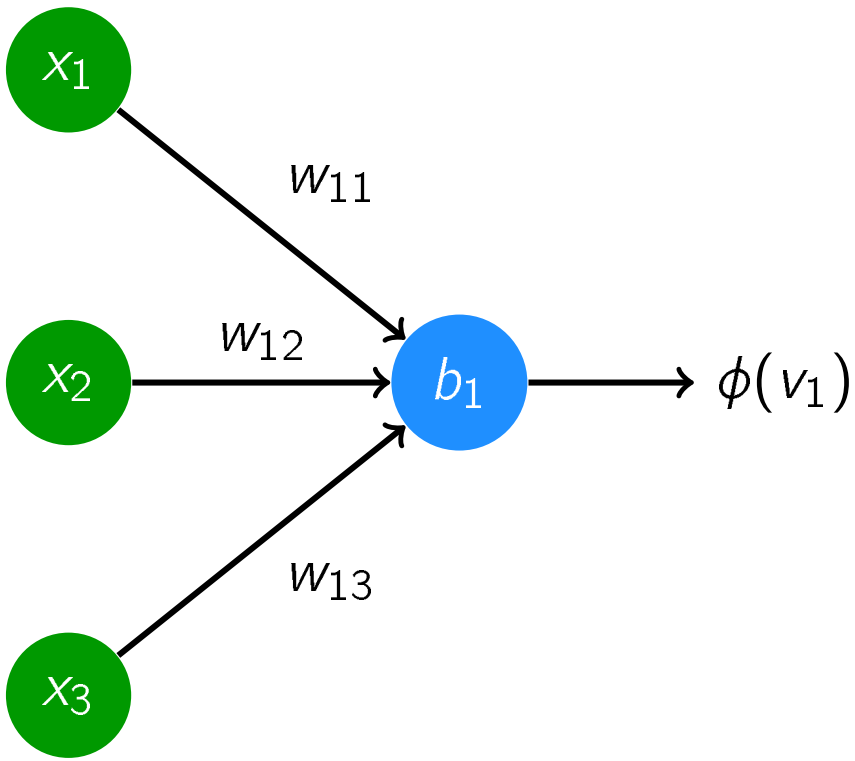
Blackbox

Supervised

Compare to data

Need large data set

Neurons



Perceptron

Input

$$x_i$$

Weight

$$w^l_{ji}$$

Bias

$$b^l_i$$

Neuron input

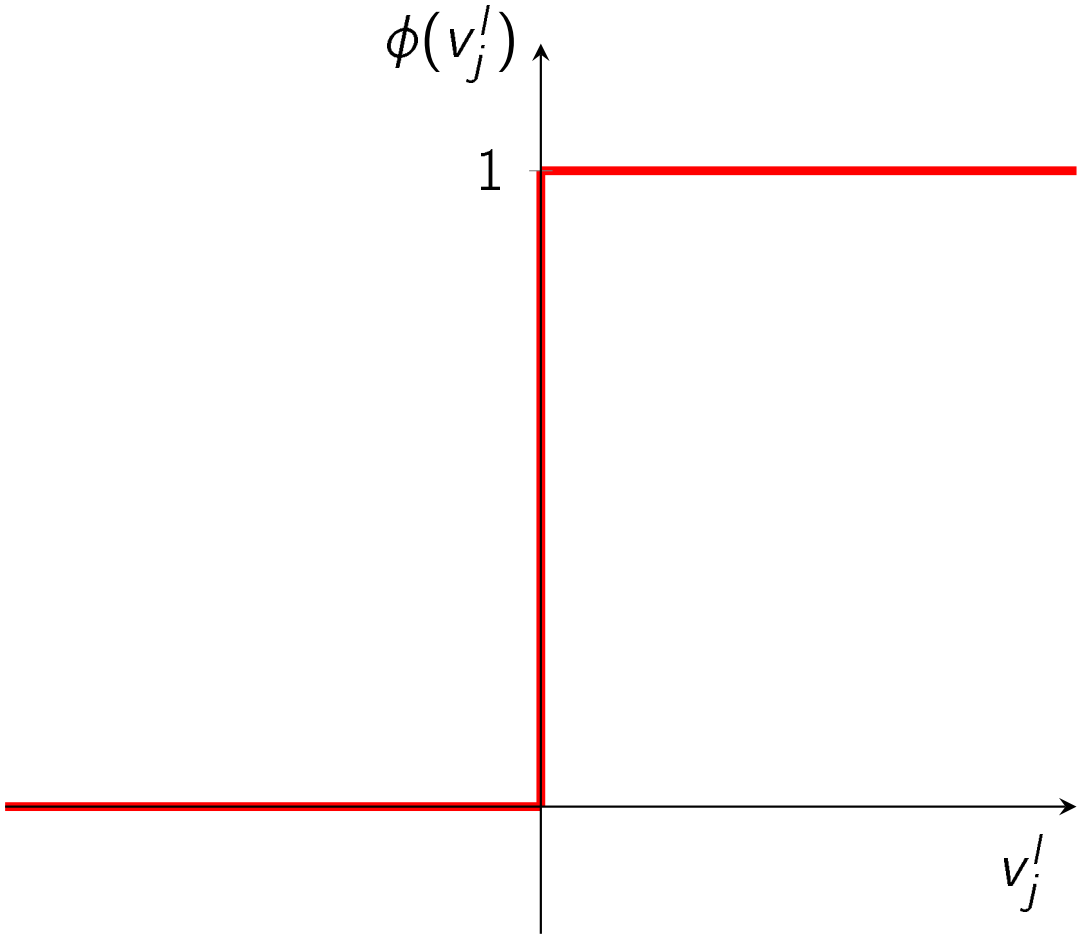
$$v^l_j = w^l_{ji} x_i + b^l_i$$

Activation

$$\phi(v^l_j) = \begin{cases} 0 & v^l_j \leq 0 \\ 1 & v^l_j > 0 \end{cases}$$

Neurons

Perceptron



Input

x_i

Weight

w_{ji}^l

Bias

b_i^l

Neuron input

$$v_j^l = w_{ji}^l x_i + b_i$$

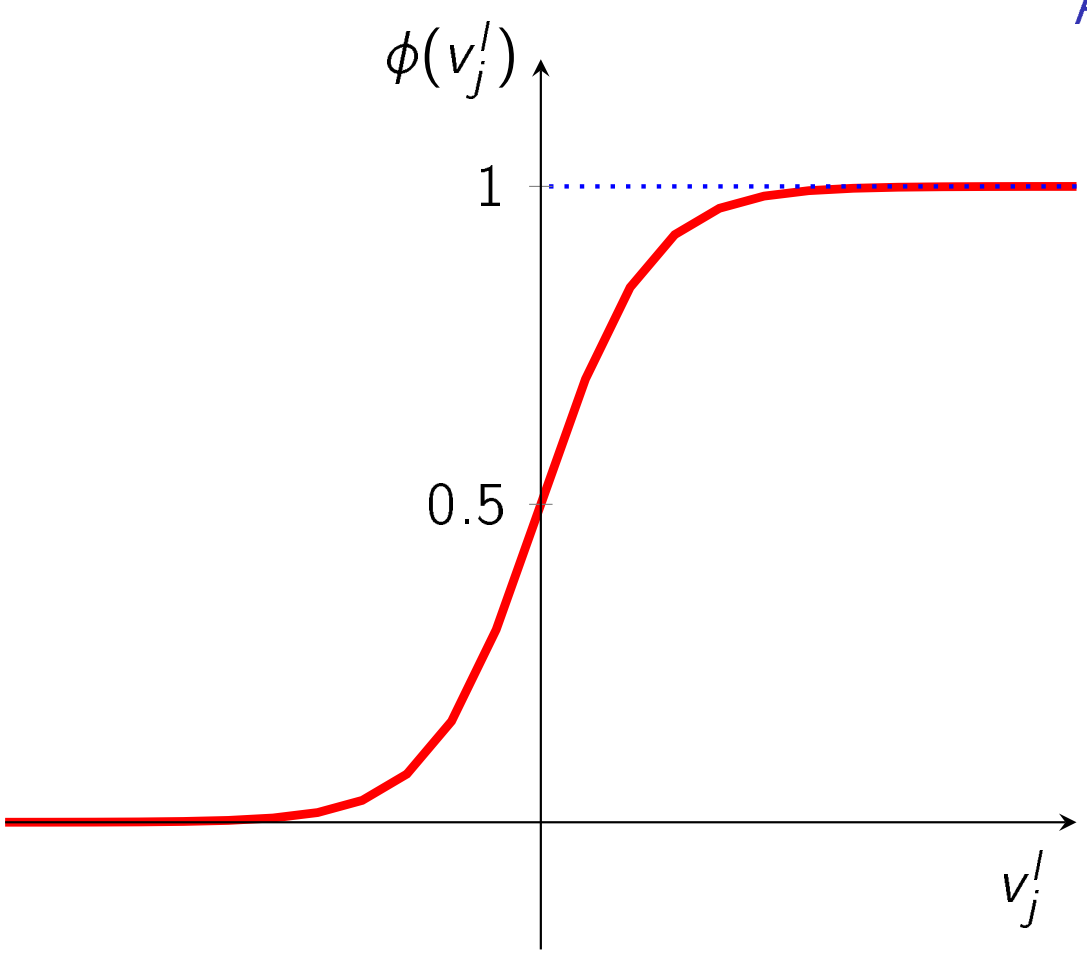
Activation

$$\phi(v_j^l) = \begin{cases} 0 & v_j^l \leq 0 \\ 1 & v_j^l > 0 \end{cases}$$

Training

$$\Delta\phi(v_j^l) \approx \frac{\partial\phi(v_j^l)}{\partial w_{ji}^l}\Delta w_{ji}^l + \frac{\partial\phi(v_j^l)}{\partial b_j^l}\Delta b_j^l$$

Activation

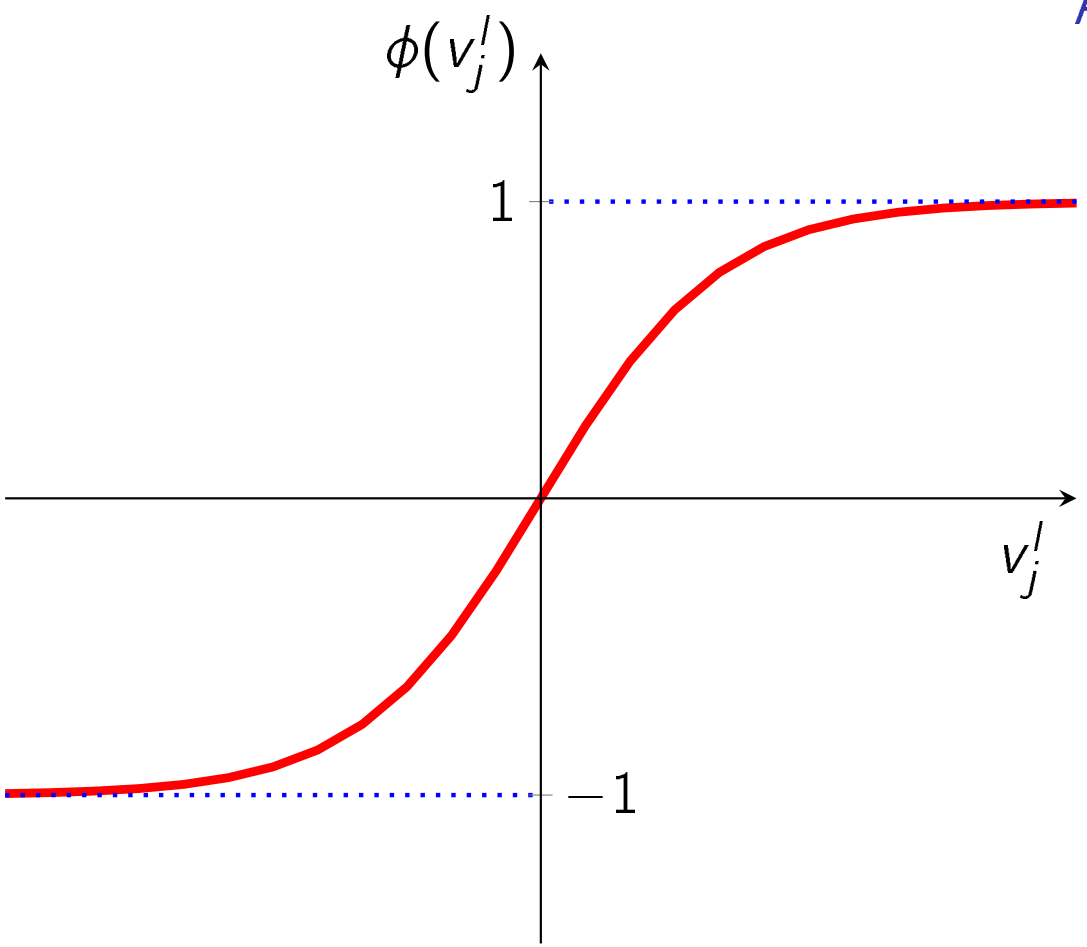


Artificial neurons

Range of activation functions $\phi(v_j^l)$

`sigmoid`

Activation

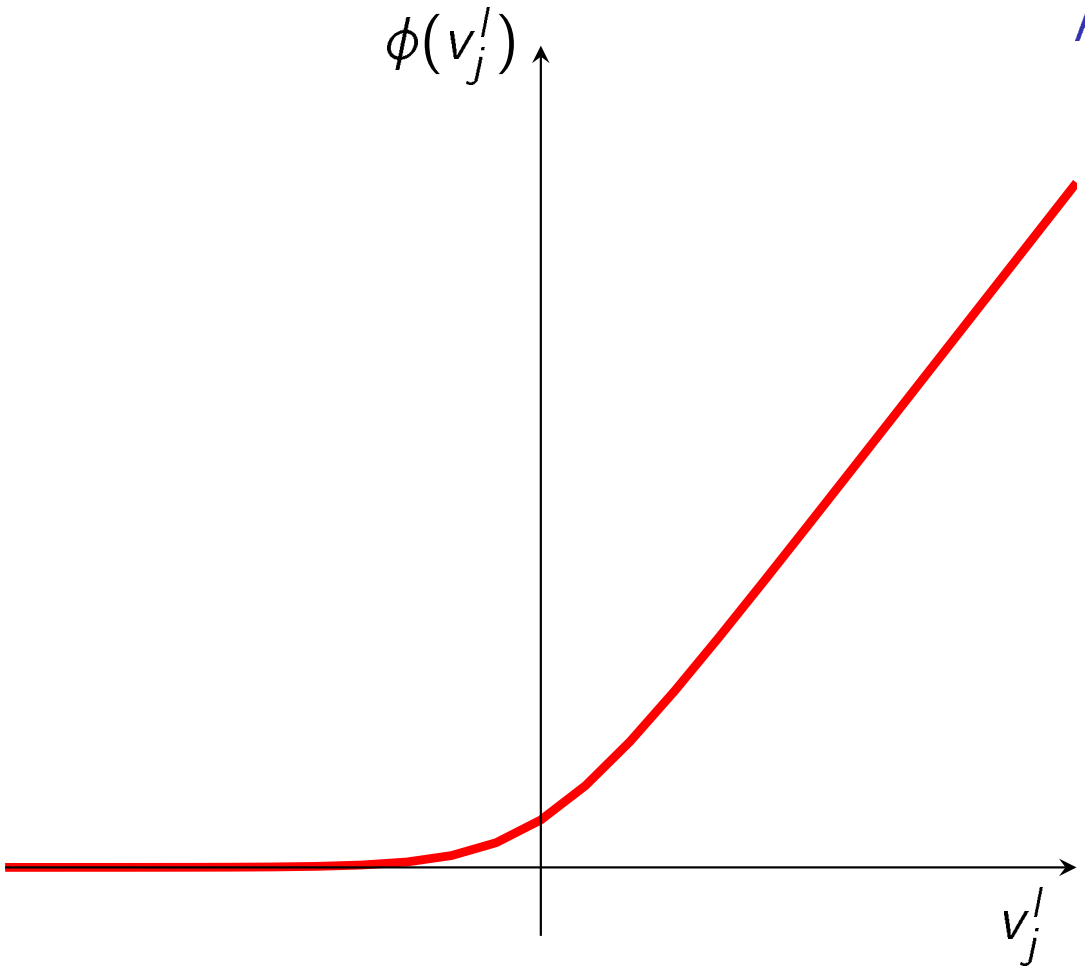


Artificial neurons

Range of activation functions $\phi(v_j^l)$

`tanh`

Activation

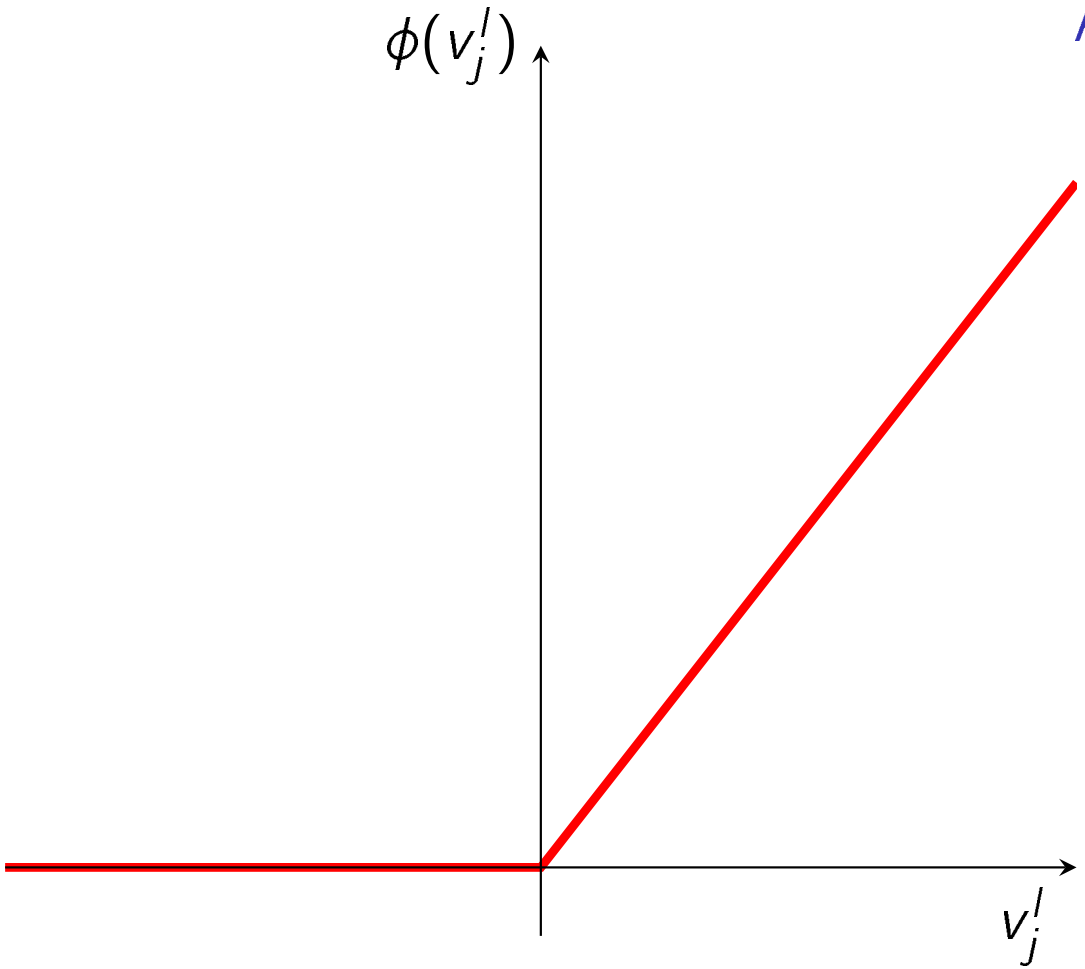


Artificial neurons

Range of activation functions $\phi(v_j^l)$

`softmax`

Activation

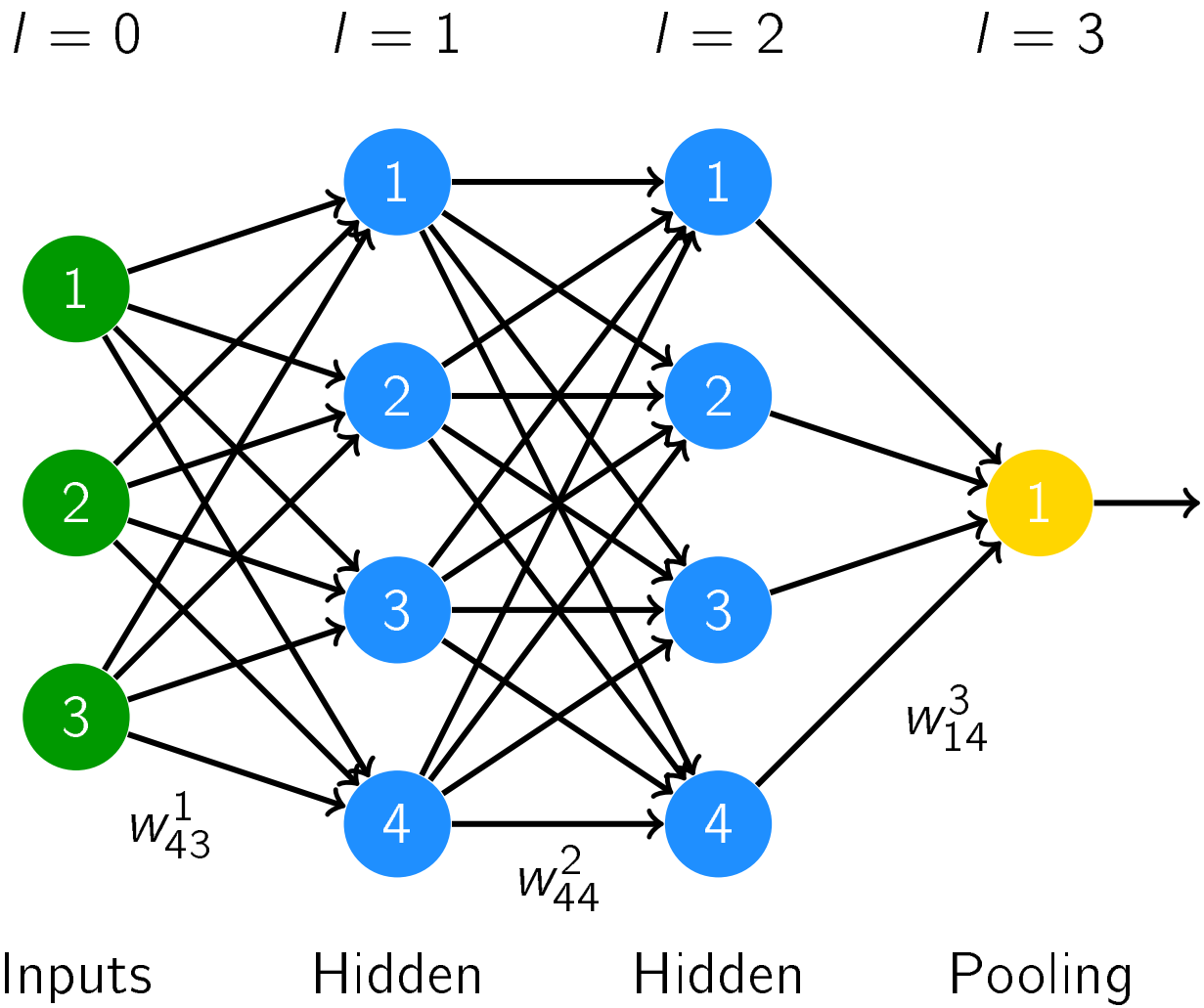


Artificial neurons

Range of activation functions $\phi(v_j^l)$

`ReLU`

Multilayer perceptrons



Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Loss function

$$\lambda(\mathbf{w}, \mathbf{b}) \quad = \quad \frac{1}{2n} \sum_{\mathrm{x}}^{n} ||\mathbf{y}(\mathbf{x}) - \phi(\mathbf{v}^L)||^2$$

Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Loss function

$$\lambda(\mathbf{w}, \mathbf{b}) \quad = \quad -\mathbf{t} \ln \phi(\mathbf{v}^L)$$

Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Loss function

$$\lambda(\mathbf{w}, \mathbf{b}) \quad = \quad -\mathbf{t} \ln \phi(\mathbf{v}^L)$$

Gradient decent

$$\Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{w}^l}\Delta\mathbf{w}^l \qquad \Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{b}^l}\Delta\mathbf{b}^l$$

Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Loss function

$$\lambda(\mathbf{w}, \mathbf{b}) \quad = \quad -\mathbf{t} \ln \phi(\mathbf{v}^L)$$

Gradient decent

$$\Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{w}^l}\Delta\mathbf{w}^l \qquad \Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{b}^l}\Delta\mathbf{b}^l$$

$$\Delta\mathbf{w}^l = -\eta\frac{\partial\lambda}{\partial\mathbf{w}^l} \qquad \Delta\mathbf{b}^l = -\eta\frac{\partial\lambda}{\partial\mathbf{b}^l}$$

Training

Output

$$\phi(v_j^l) \quad = \quad \phi(w_{ji}^l \phi(v_i^{l-1}) + b_j^l)$$

Loss function

$$\lambda(\mathbf{w}, \mathbf{b}) \quad = \quad -\mathbf{t} \ln \phi(\mathbf{v}^L)$$

Gradient decent

$$\Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{w}^l}\Delta\mathbf{w}^l \qquad \Delta\lambda \approx \frac{\partial\lambda}{\partial\mathbf{b}^l}\Delta\mathbf{b}^l$$

$$\Delta\mathbf{w}^l = -\eta\frac{\partial\lambda}{\partial\mathbf{w}^l} \qquad \Delta\mathbf{b}^l = -\eta\frac{\partial\lambda}{\partial\mathbf{b}^l}$$

$$\Delta\lambda \approx -\eta||\partial\lambda/\partial\mathbf{w}^l||^2 \leq 0 \qquad \Delta\lambda \approx -\eta||\partial\lambda/\partial\mathbf{b}^l||^2 \leq 0$$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$

2. Calculate output at each layer

$$\phi(\mathbf{v}^l) = \phi(\mathbf{w}^l \phi(\mathbf{v}^{l-1}) + \mathbf{b}^l) \; \forall l$$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$
2. Calculate output at each layer
$$\phi(\mathbf{v}^l) = \phi(\mathbf{w}^l \phi(\mathbf{v}^{l-1}) + \mathbf{b}^l) \; \forall l$$
3. Compute change in loss function
$$\frac{\partial \lambda}{\partial \mathbf{v}^L} = \frac{\partial \lambda}{\partial \phi(\mathbf{v}^L)} \odot \frac{\partial \phi(\mathbf{v}^L)}{\partial \mathbf{v}^L}$$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$
2. Calculate output at each layer
$$\phi(\mathbf{v}^l) = \phi(\mathbf{w}^l \phi(\mathbf{v}^{l-1}) + \mathbf{b}^l) \; \forall l$$
3. Compute change in loss function
$$\frac{\partial \lambda}{\partial \mathbf{v}^L} = \frac{\partial \lambda}{\partial \phi(\mathbf{v}^L)} \odot \frac{\partial \phi(\mathbf{v}^L)}{\partial \mathbf{v}^L}$$
4. Back propagate through network
$$\frac{\partial \lambda}{\partial \mathbf{v}^l} = \left(\mathbf{w}^{l+1}\right)^T \frac{\partial \lambda}{\partial \mathbf{v}^{l+1}} \odot \frac{\partial \phi(\mathbf{v}^l)}{\partial \mathbf{v}^l} \; \forall l$$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$
2. Calculate output at each layer
$$\phi(\mathbf{v}^l) = \phi(\mathbf{w}^l \phi(\mathbf{v}^{l-1}) + \mathbf{b}^l) \; \forall l$$
3. Compute change in loss function
$$\frac{\partial \lambda}{\partial \mathbf{v}^L} = \frac{\partial \lambda}{\partial \phi(\mathbf{v}^L)} \odot \frac{\partial \phi(\mathbf{v}^L)}{\partial \mathbf{v}^L}$$
4. Back propagate through network
$$\frac{\partial \lambda}{\partial \mathbf{v}^l} = \left(\mathbf{w}^{l+1}\right)^T \frac{\partial \lambda}{\partial \mathbf{v}^{l+1}} \odot \frac{\partial \phi(\mathbf{v}^l)}{\partial \mathbf{v}^l} \; \forall l$$
5. Find gradients with respect to weights and biases
$$\frac{\partial \lambda}{\partial \mathbf{v}^l} = \frac{1}{\phi(\mathbf{v}^{l-1})} \frac{\partial \lambda}{\partial \mathbf{w}^l} \text{ and } \frac{\partial \lambda}{\partial \mathbf{v}^l} = \frac{\partial \lambda}{\partial \mathbf{b}^l} \; \forall l$$

Training

1. Input data $\mathbf{x} = \phi(\mathbf{v}^0)$

2. Calculate output at each layer
$$\phi(\mathbf{v}^l) = \phi(\mathbf{w}^l \phi(\mathbf{v}^{l-1}) + \mathbf{b}^l) \; \forall l$$

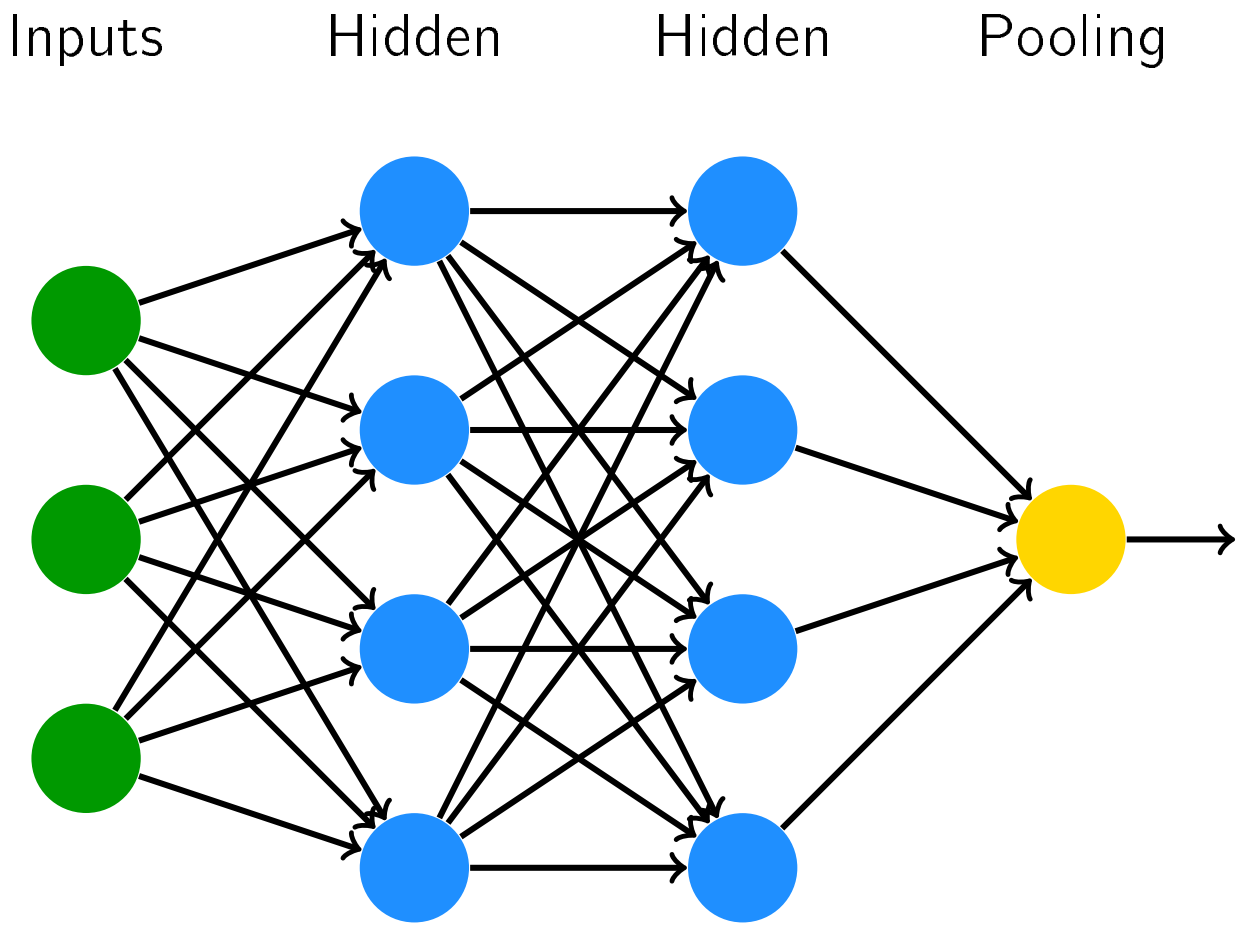
3. Compute change in loss function
$$\frac{\partial \lambda}{\partial \mathbf{v}^L} = \frac{\partial \lambda}{\partial \phi(\mathbf{v}^L)} \odot \frac{\partial \phi(\mathbf{v}^L)}{\partial \mathbf{v}^L}$$

4. Back propagate through network
$$\frac{\partial \lambda}{\partial \mathbf{v}^l} = \left(\mathbf{w}^{l+1}\right)^T \frac{\partial \lambda}{\partial \mathbf{v}^{l+1}} \odot \frac{\partial \phi(\mathbf{v}^l)}{\partial \mathbf{v}^l} \; \forall l$$

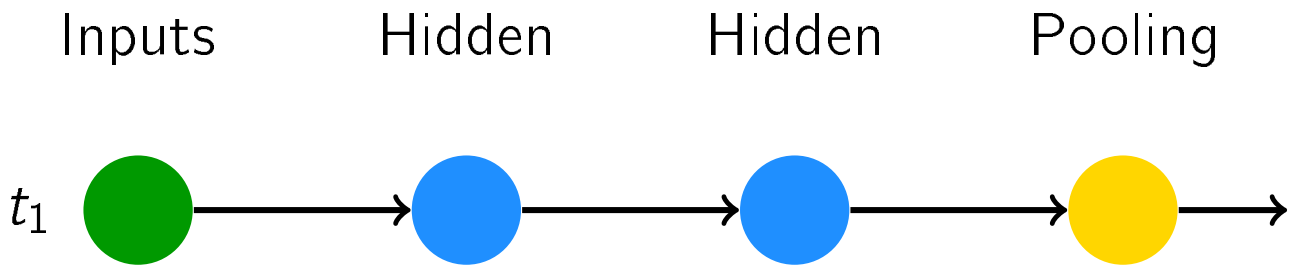
5. Find gradients with respect to weights and biases
$$\frac{\partial \lambda}{\partial \mathbf{v}^l} = \frac{1}{\phi(\mathbf{v}^{l-1})} \frac{\partial \lambda}{\partial \mathbf{w}^l} \text{ and } \frac{\partial \lambda}{\partial \mathbf{v}^l} = \frac{\partial \lambda}{\partial \mathbf{b}^l} \; \forall l$$

6. Update weights and biases
$$\mathbf{w}^l \rightarrow \mathbf{w}^l - \eta \frac{\partial \lambda}{\partial \mathbf{w}^l} \text{ and } \mathbf{b}^l \rightarrow \mathbf{b}^l - \eta \frac{\partial \lambda}{\partial \mathbf{b}^l} \; \forall l$$

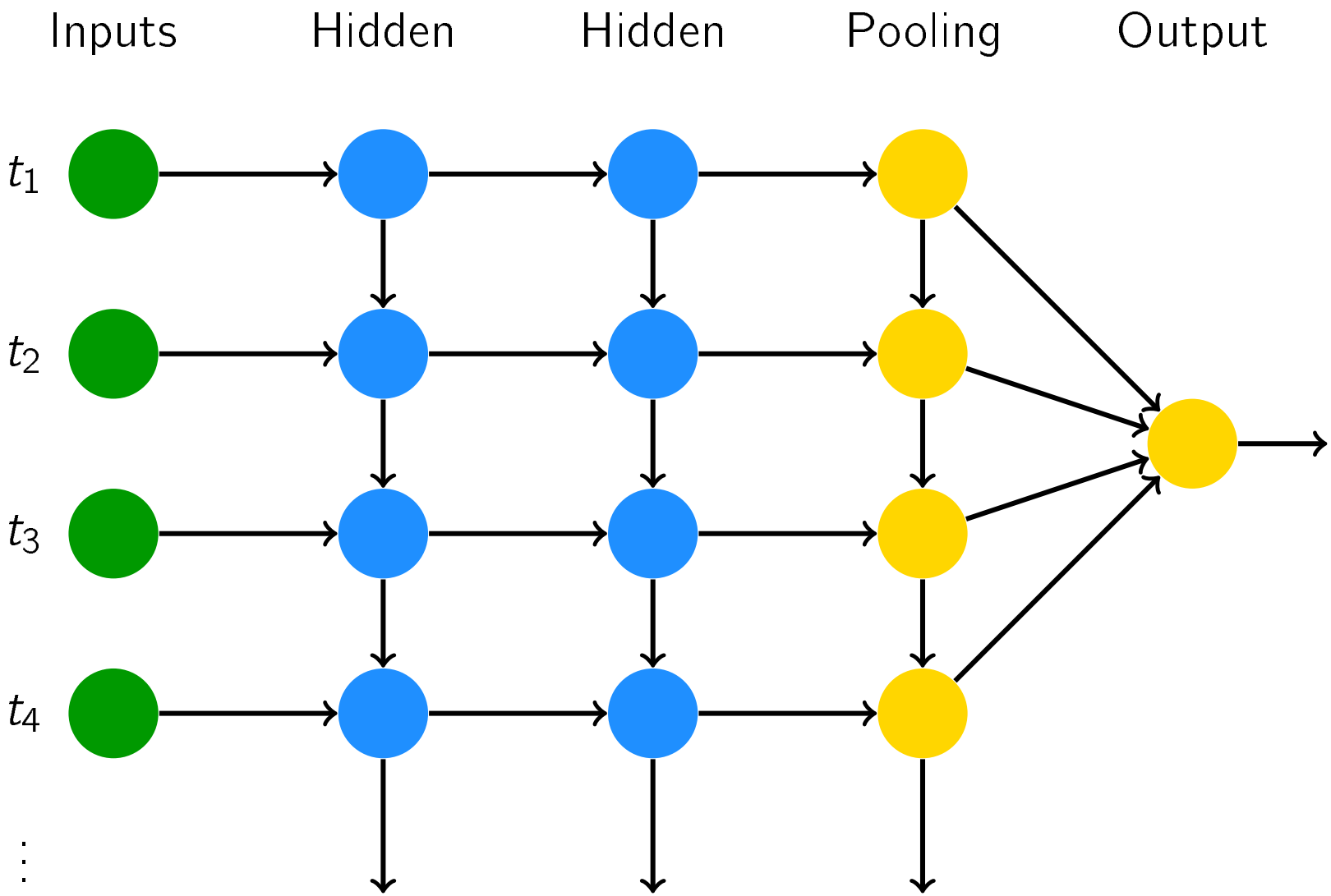
Recurrent neural networks



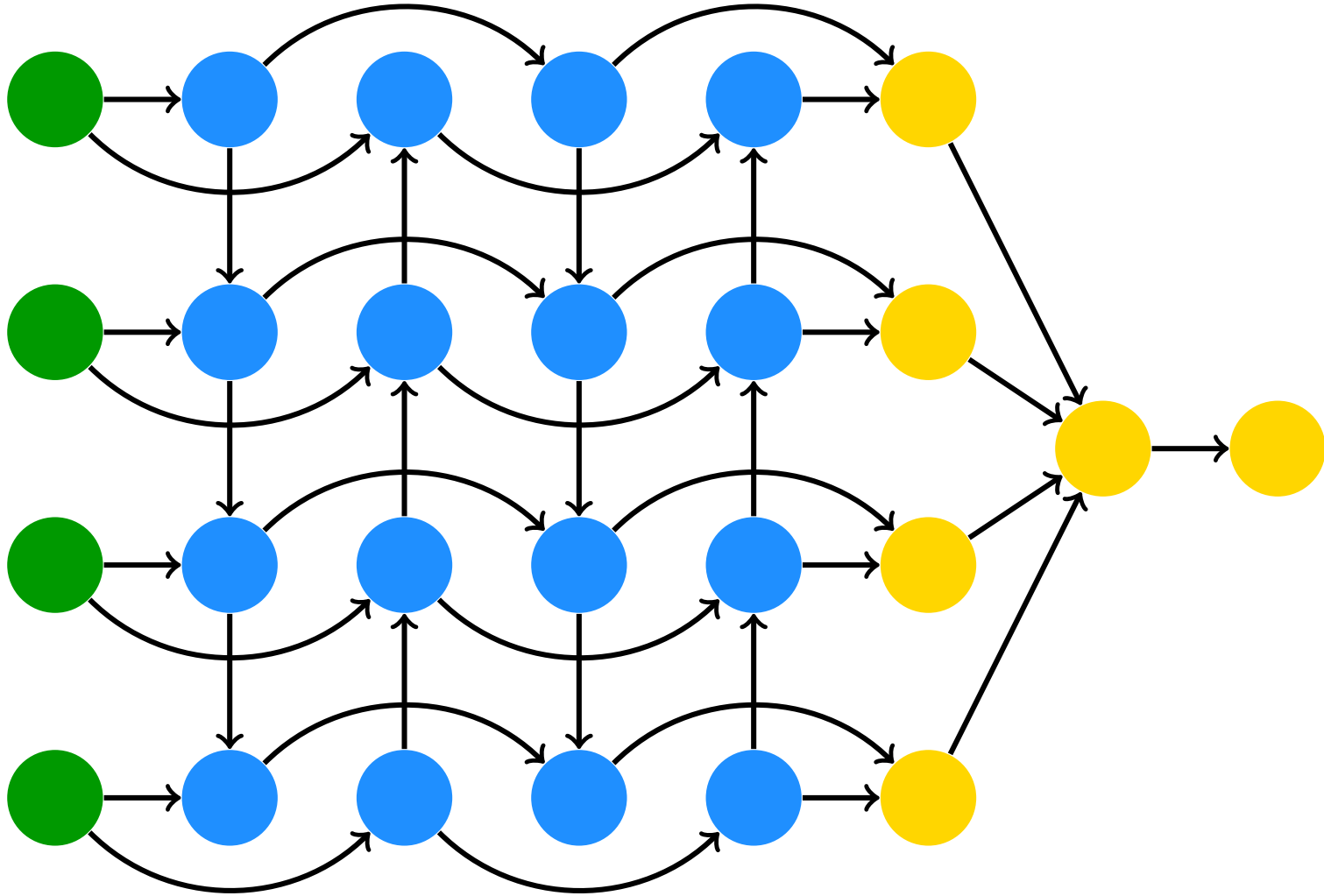
Recurrent neural networks



Recurrent neural networks



Recurrent neural networks



Inputs Hidden left Hidden right Hidden left Hidden right Merge Pooling (Mean) Output (`softmax`)

Uses of recurrent neural networks

Time-domain sequences

- Natural language processing
- Speech recognition

Uses of recurrent neural networks

Time-domain sequences

- Natural language processing
- Speech recognition

Cosmology and astronomy

- Classification of supernovae

Supernovae photometric classification challenge

Supernovae

Simulated DES light curves

Types I, II, III, etc.

21319 light curves

Training set $\sim$ 1000

Supernovae photometric classification challenge

Supernovae

Simulated DES light curves

Types I, II, III, etc.

21319 light curves

Training set ~ 1000

Data

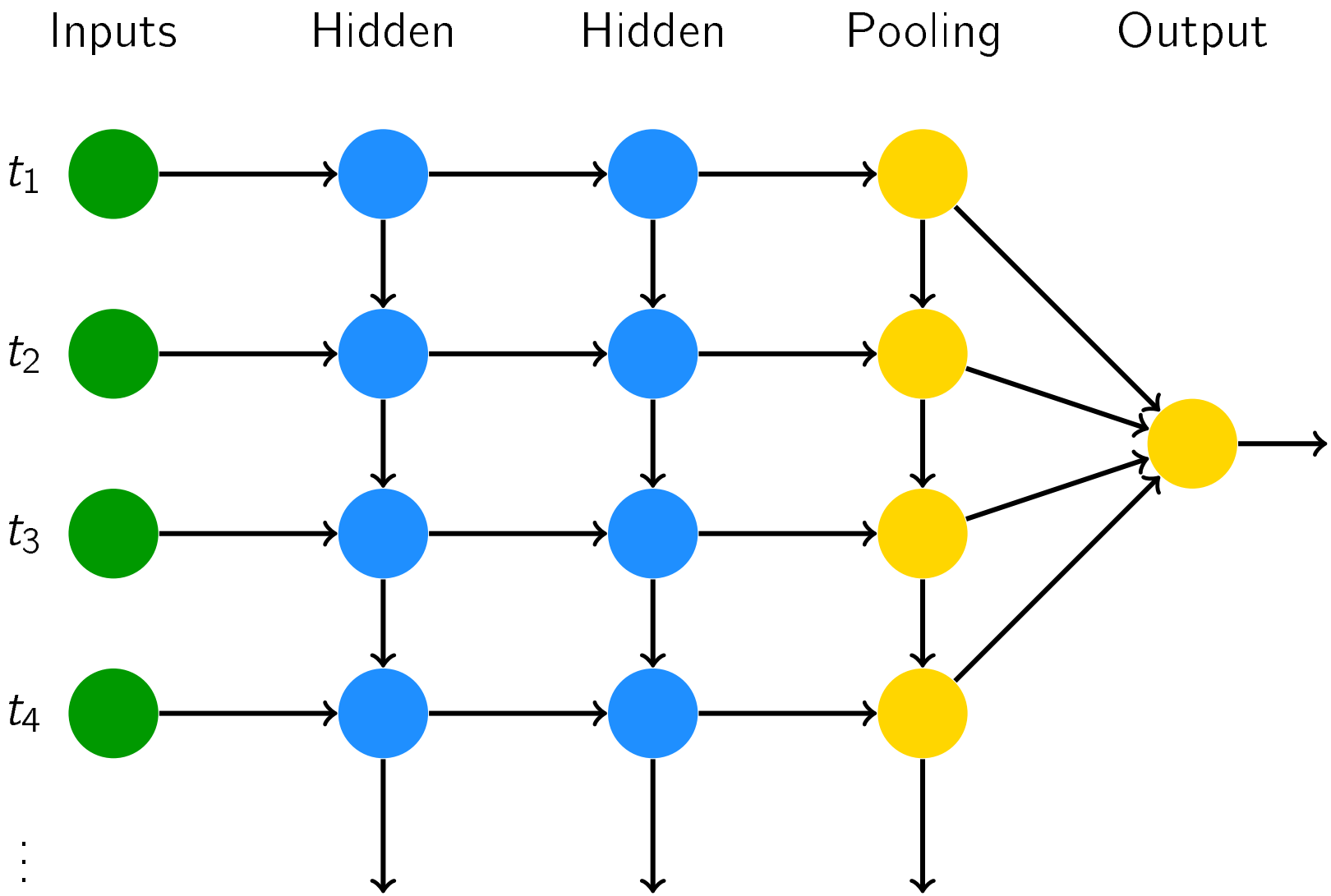
Fluxes (g, r, i, z) and errors

Position (RA and Dec)

Dust extinction

z of host galaxy

Classifying supernovae with recurrent neural networks



Supernovae photometric classification challenge

1. Type Ia vs. non-Type Ia

Supernovae photometric classification challenge

1. Type Ia vs. non-Type Ia
2. Type I vs. Type II vs. Type III

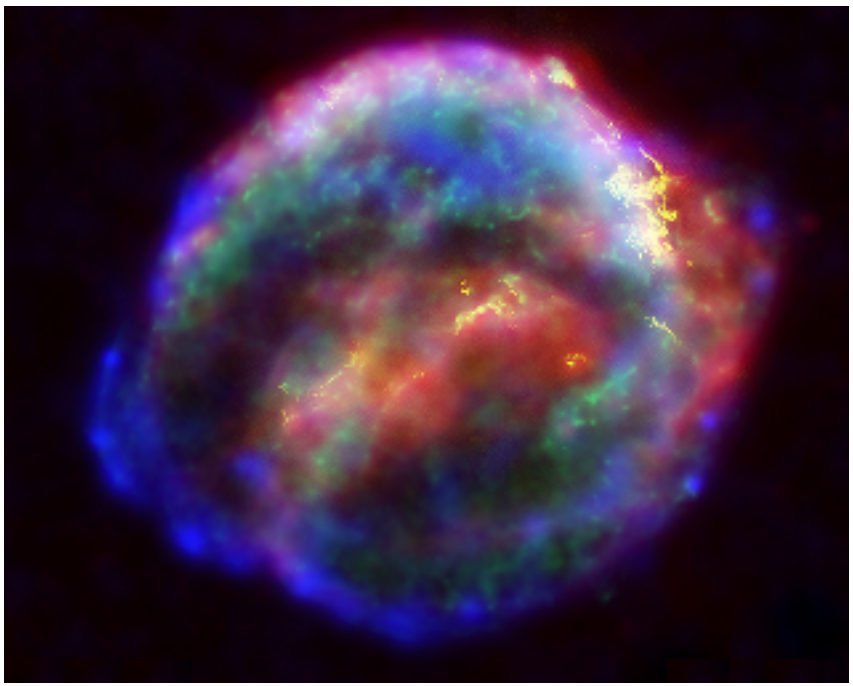
Supernovae photometric classification challenge

1. Type Ia vs. non-Type Ia
2. Type I vs. Type II vs. Type III
3. Type Ia vs. non-Type Ia (early epoch data)

Supernovae photometric classification challenge

1. Type Ia vs. non-Type Ia
2. Type I vs. Type II vs. Type III
3. Type Ia vs. non-Type Ia (early epoch data)
4. Type I vs. Type II vs. Type III (early epoch data)

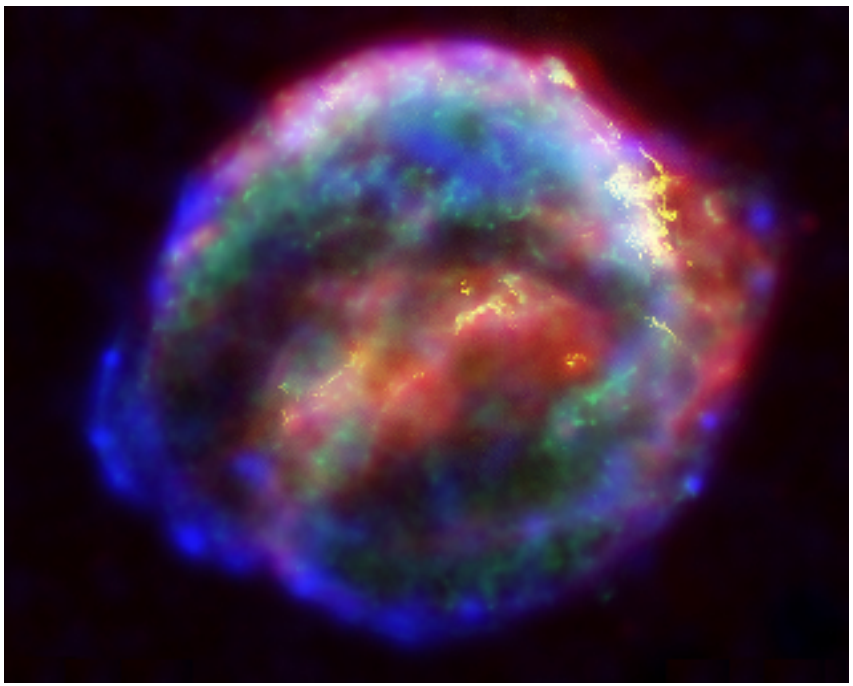
Supernovae photometric classification challenge



Accuracy

$$\text{Accuracy} = \frac{\text{TP}}{\text{Total}}$$

Supernovae photometric classification challenge



Accuracy

$$\text{Accuracy} = \frac{\text{TP}}{\text{Total}}$$

Figure of merit

$$F_1 = \frac{1}{\text{TP} + \text{FN}} \frac{\text{TP}^2}{\text{TP} + 3 \times \text{FP}}$$

How others did

SALT2

Spectral Adaptive Lightcurve Template

$$\text{Accuracy} = 96\% \quad \text{and} \quad F_1 \sim 0.6$$

How others did

SALT2

Spectral Adaptive Lightcurve Template

$$\text{Accuracy} = 96\% \quad \text{and} \quad F_1 \sim 0.6$$

Machine learning techniques

Method	Train	Accuracy	F_1
Karpenka et al. (2013)	1,045		0.33
Karpenka et al. (2013)	10,660		0.58
Newling et al. (2011)	2,000		0.45
Newling et al. (2011)	8,000		0.55
Lochner et al. (2016)	1,103	0.94 ± 0.03	

How we did

1. Type Ia vs. non-Type Ia

Unidirectional LSTM, [4] with 0.5 dropout

Train	Accuracy	F_1
1,103	85.9 $\pm$ 0.9	0.31$\pm$0.03

How we did

1. Type Ia vs. non-Type Ia

Unidirectional LSTM, [4] with 0.5 dropout

Train	Accuracy	F_1
1,103	85.9 ± 0.9	0.31±0.03

Unidirectional LSTM, [16, 16] with 0.5 dropout

Train	Accuracy	F_1
5,330	92.9 ± 0.6	0.57±0.03
10,660	94.7 ± 0.2	0.64±0.01

Why we were a little better

2. Type I vs. Type II vs. Type III

Bidirectional LSTM, [4] with 0.5 dropout

Train	Accuracy
1,103	78.1 ± 0.9

Bidirectional LSTM, [16, 16] with 0.5 dropout

Train	Accuracy
10,660	90.4 ± 0.3

Why we were a quite a bit better

3. Type Ia vs. non-Type Ia (early epoch data only)

Unidirectional LSTM, [4] with 0.5 dropout

Train	Accuracy	F_1
1,103	85.2 ± 1.2	0.29 ± 0.04

Unidirectional LSTM, [16, 16] with 0.5 dropout

Train	Accuracy	F_1
10,660	93.1 ± 0.4	0.58 ± 0.01

Why we are even better still

4. Type I vs. Type II vs. Type III (early epoch only data)

Unidirectional LSTM, [4] with 0.5 dropout

Train	Accuracy
1,103	76.8 ± 1.3

Unidirectional LSTM, [16, 16] with 0.5 dropout

Train	Accuracy
10,660	87.9 ± 0.9

What is most impressive

Conclusions

Supervised deep learning

How recurrent neural networks can classify supernovae

Some results