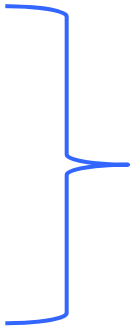
Today

- Neural networks and deep learning
  - Single layer neural networks
    - using keras package
    - architectures for different questions
      - regression, classification, multifunction
  - Multi-layer neural networks
  - Convolutional neural networks
  - Recurrent neural networks



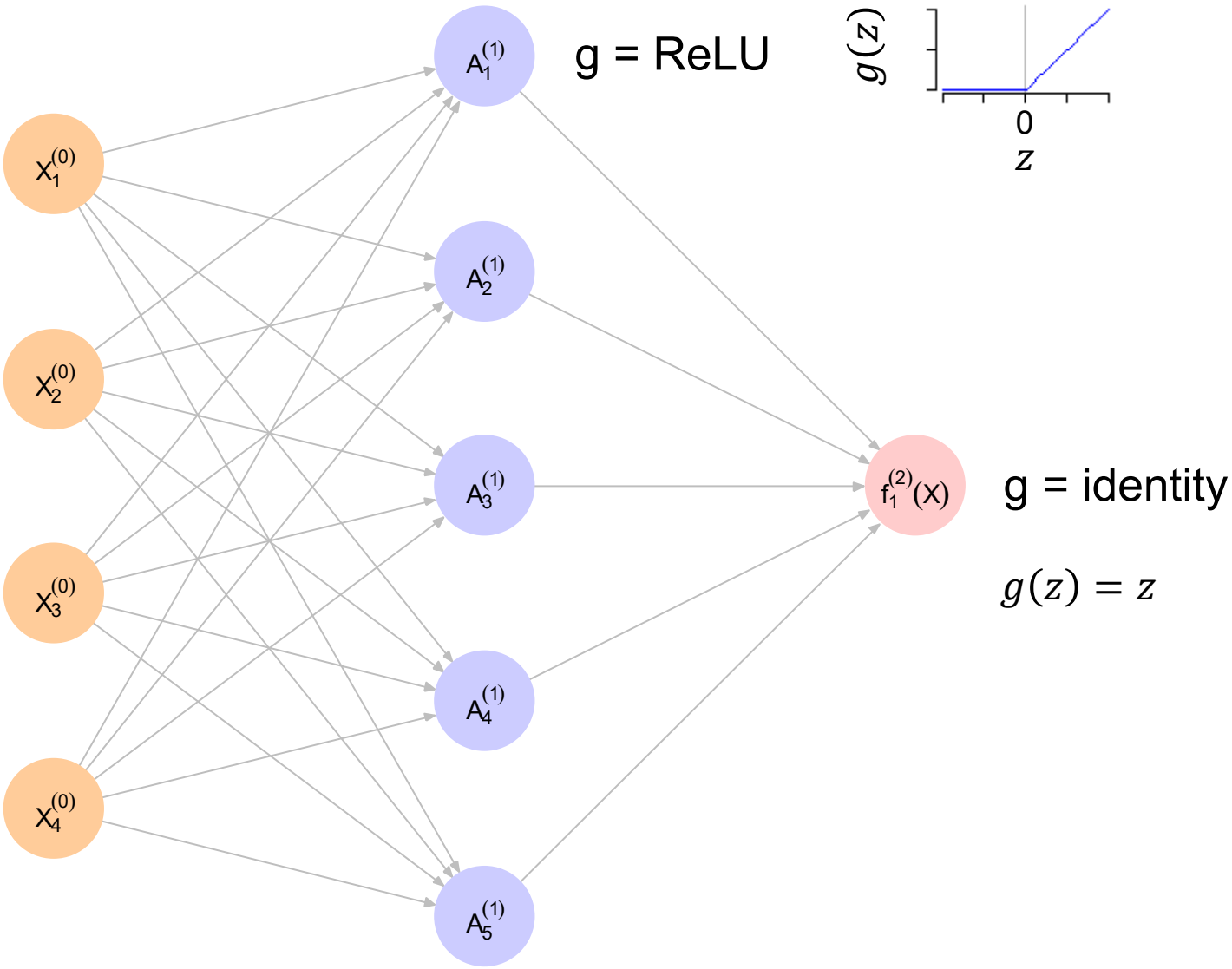
Reminder

- Data science framework
  - model algorithm
  - training algorithm
  - inference algorithm

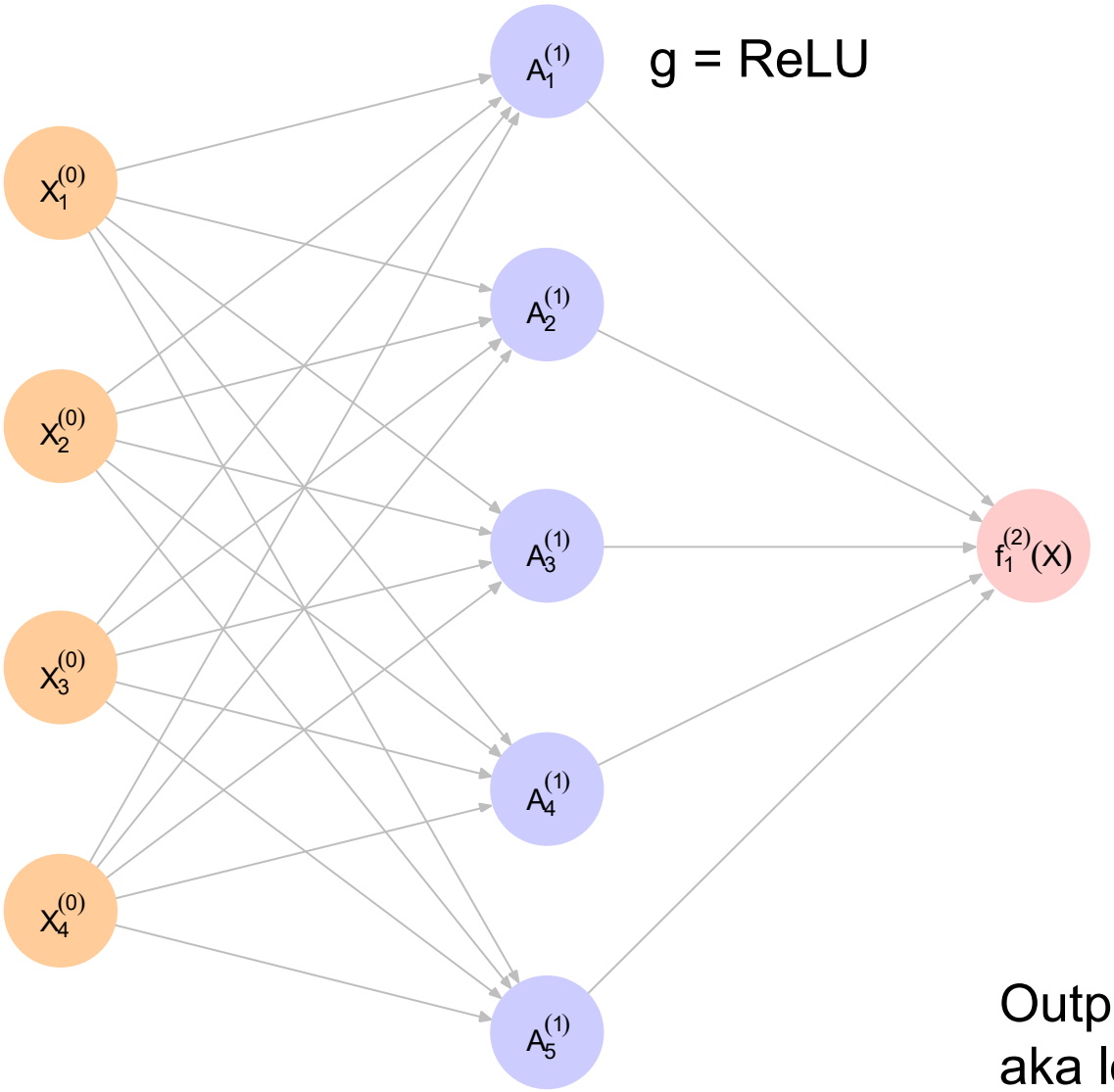
Model algorithm: architecture

- How many units and how they are connected
- What activation functions are applied to each unit

Architecture: regression

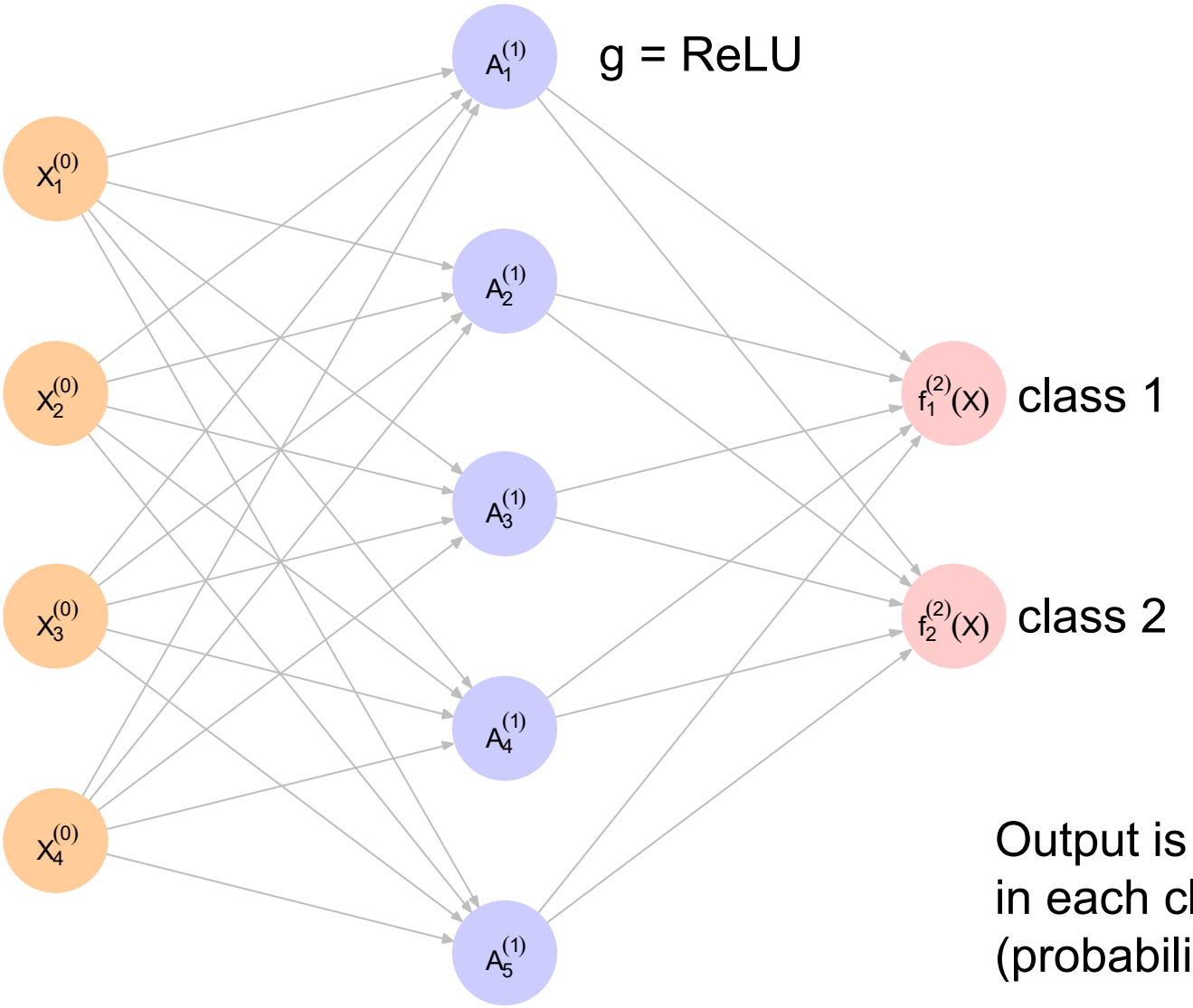


Architecture: binary classification (opt 1)



Output is probability
aka logistic regression

Architecture: binary classification (opt 2)

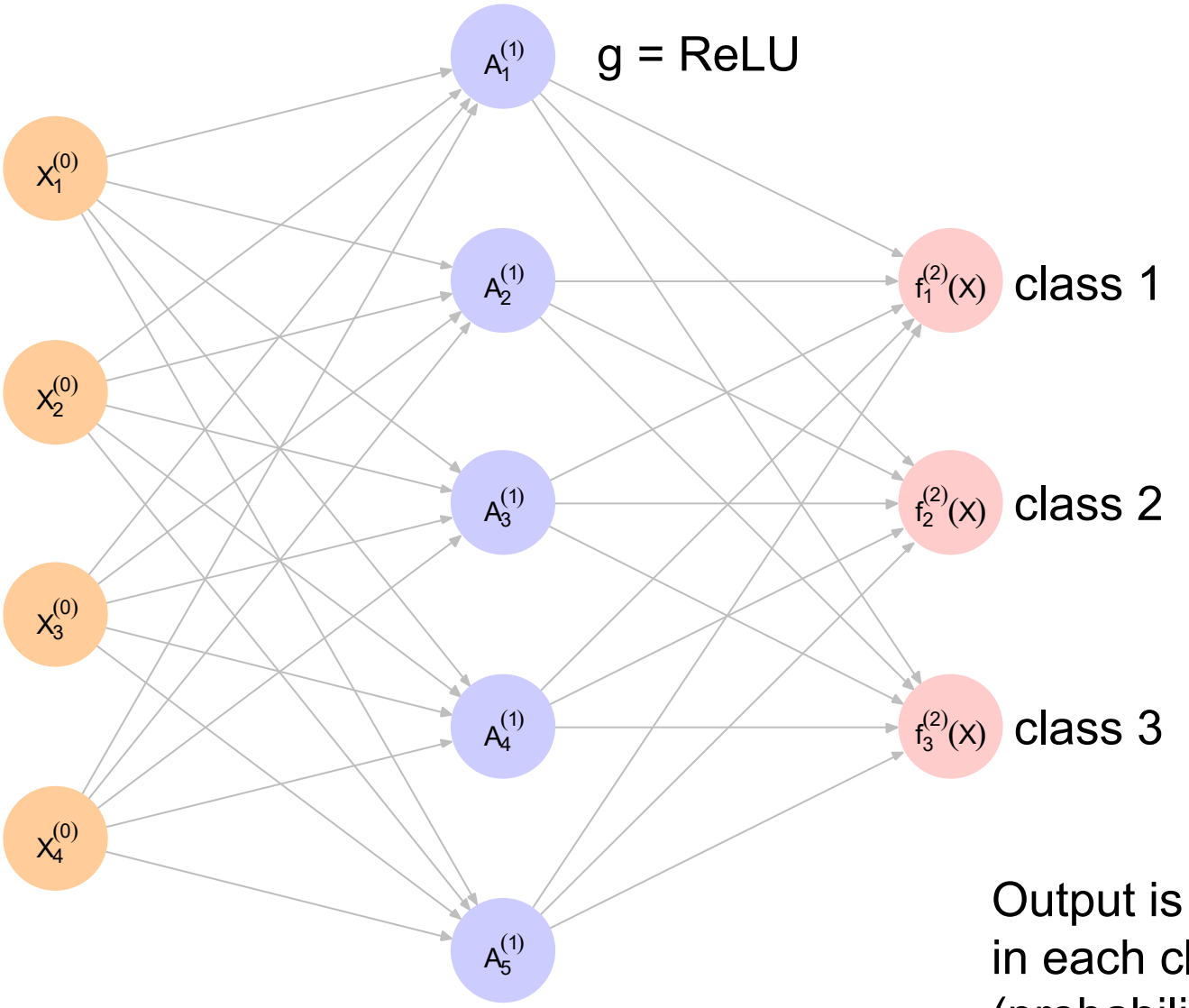


g = softmax

$$g_m(z) = \frac{e^{z_m}}{\sum_{l=1}^{K} e^{z_l}}$$

Output is probability
in each class
(probability sums to 1)

Architecture: multicategory classification

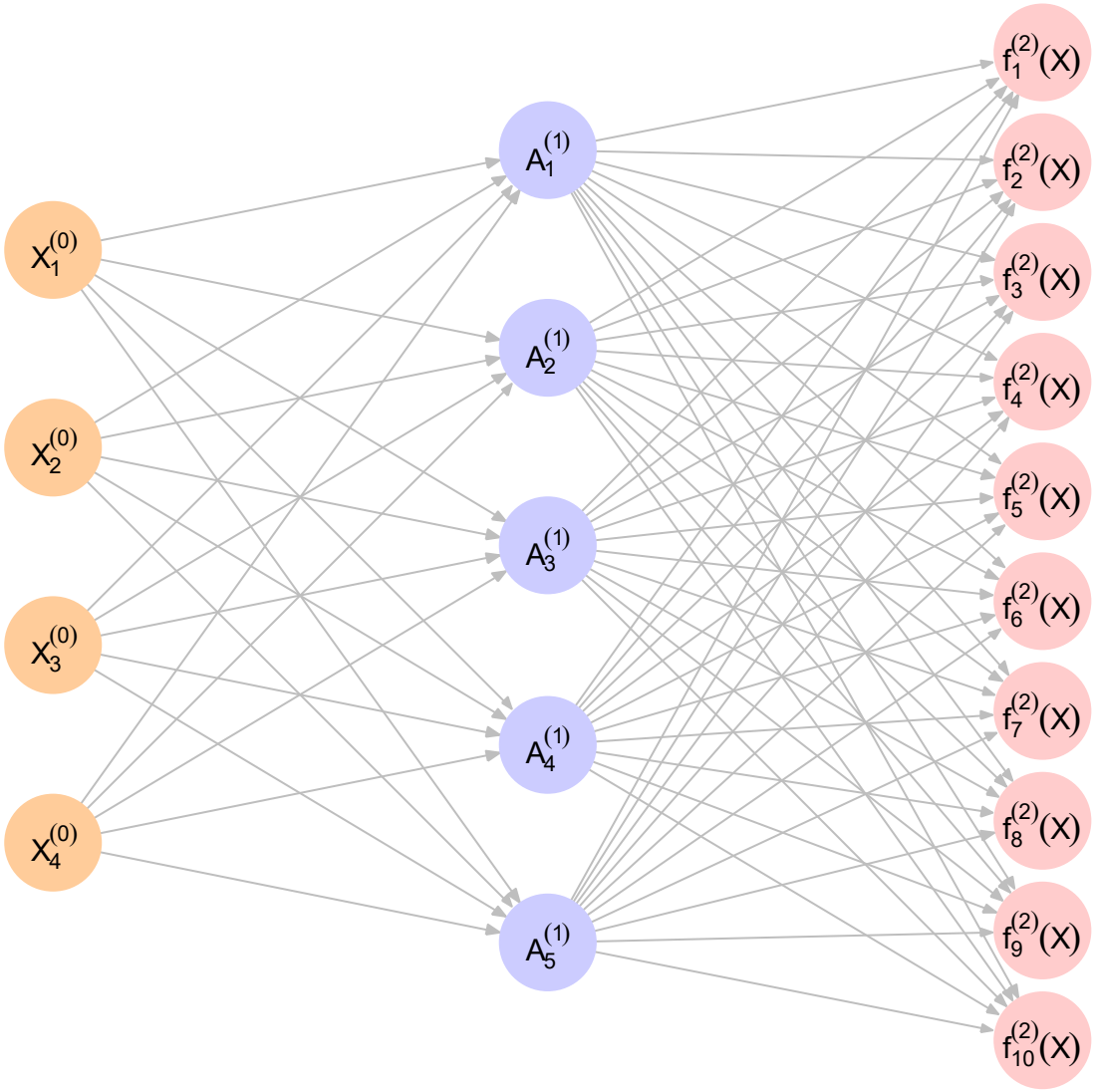


g = softmax

$$g_m(z) = \frac{e^{z_m}}{\sum_{l=1}^{K} e^{z_l}}$$

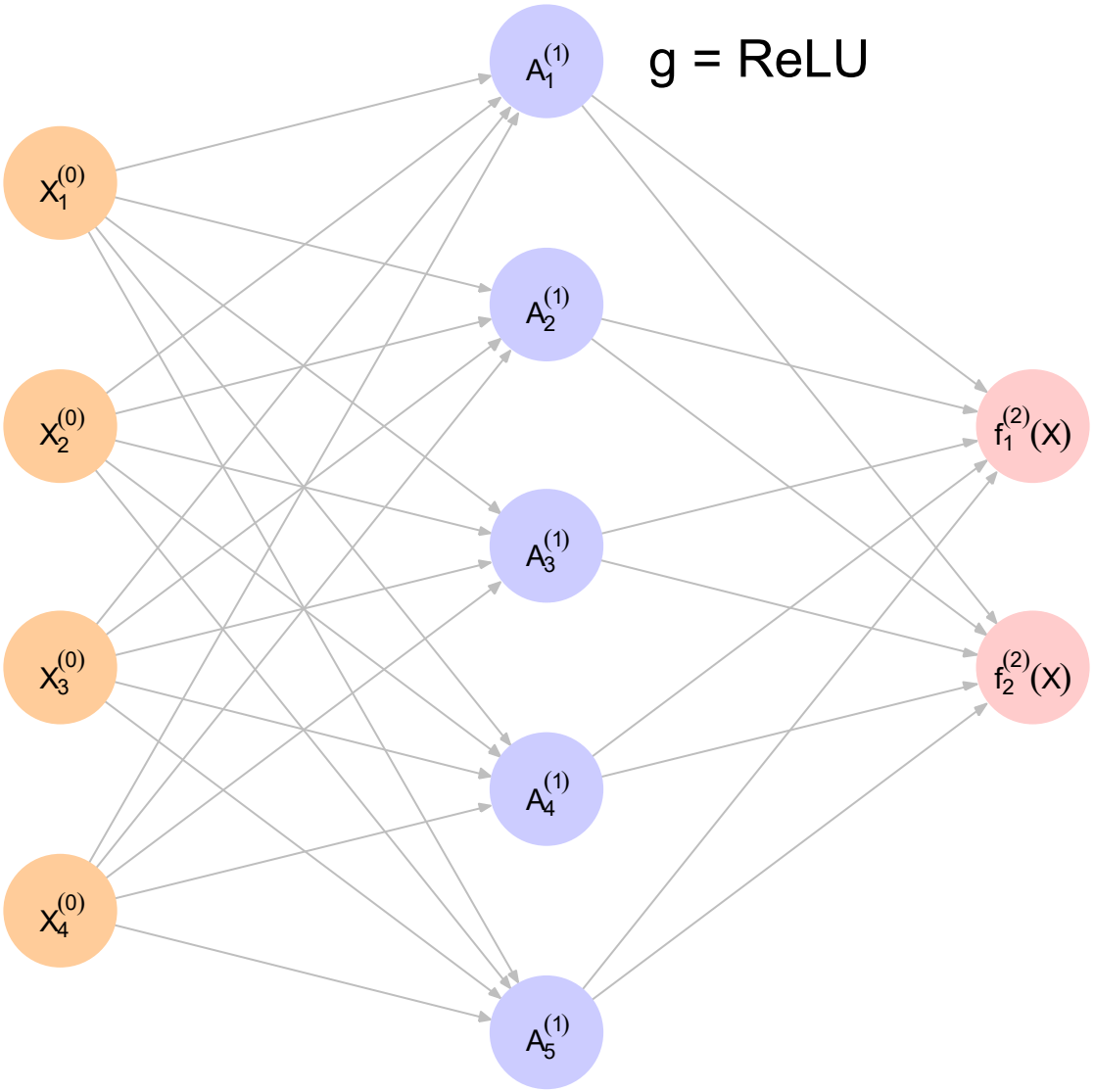
Output is probability
in each class
(probability sums to 1)

Architecture: multicategory classification



Could have any number of categories

Architecture: multifunction (examples)



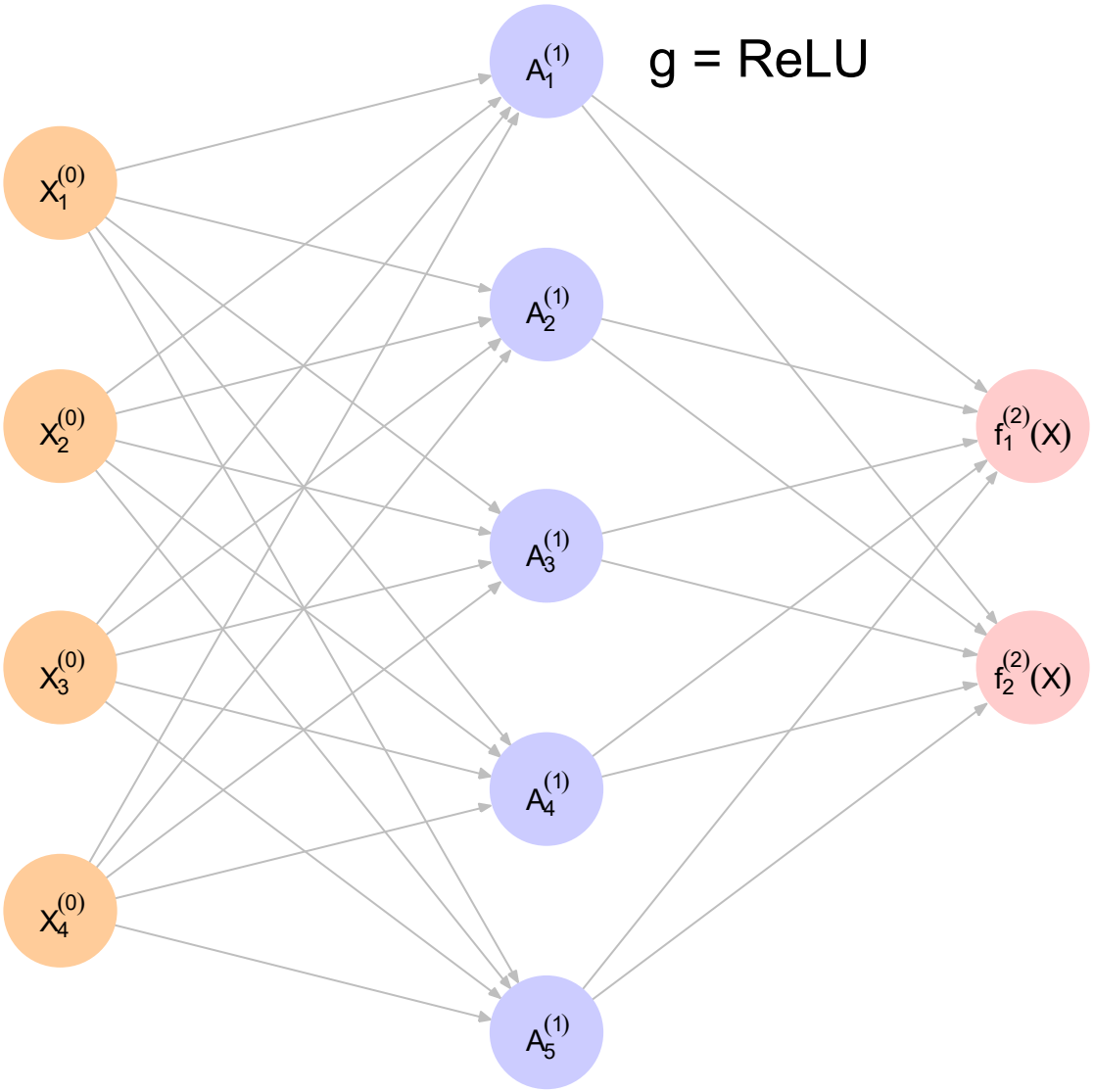
Joint species abundance model for 2 species

g = identity
Y1 = abun species 1

g = identity
Y2 = abun species 2

Output is a prediction for two different outcome variables Y1, Y2

Architecture: multifunction (examples)



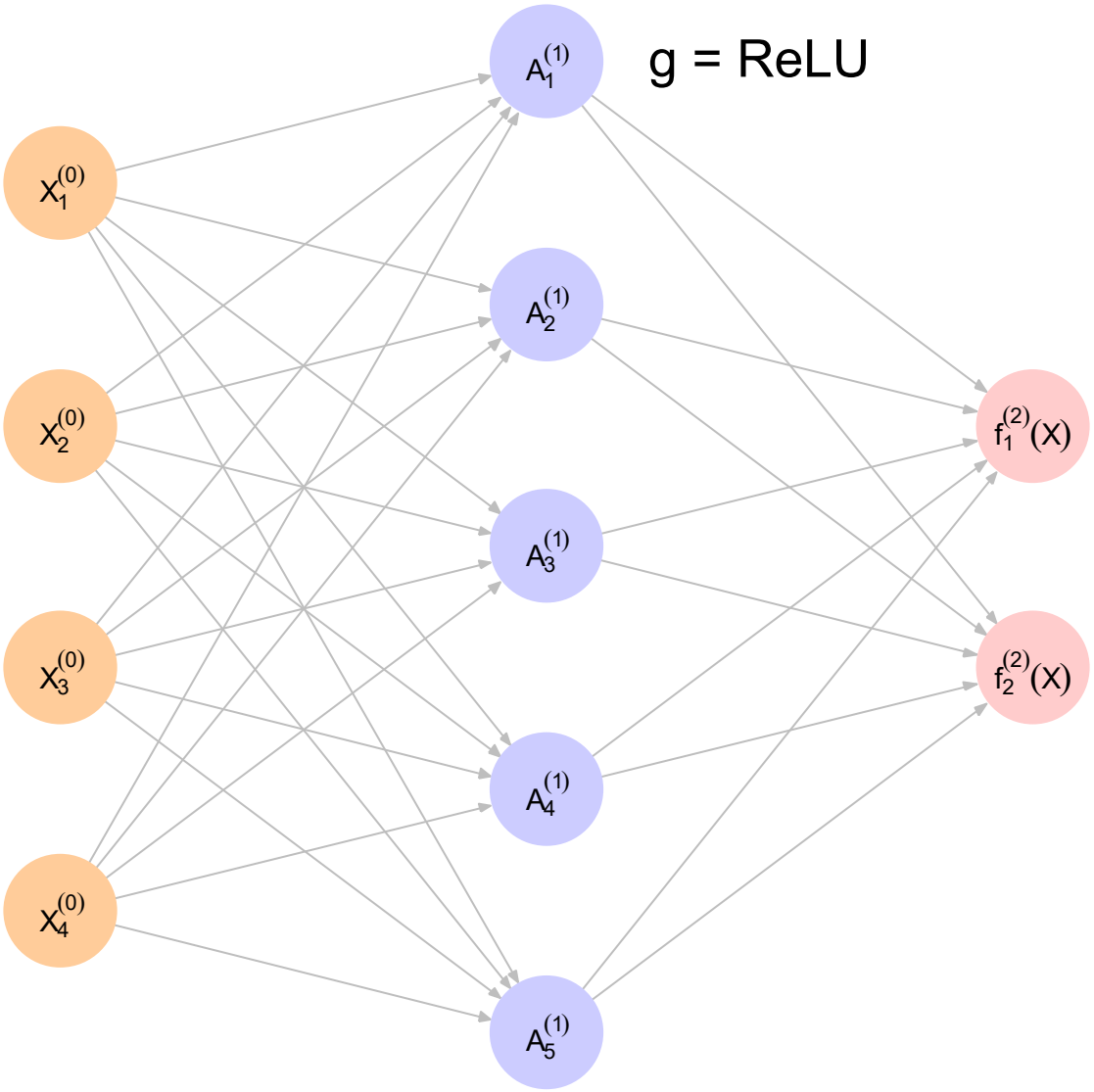
Joint species distribution model for 2 species

g = sigmoid
Y1 = prob species 1

g = sigmoid
Y2 = prob species 2

Output is a prediction for two different outcome variables Y1, Y2

Architecture: multifunction (examples)



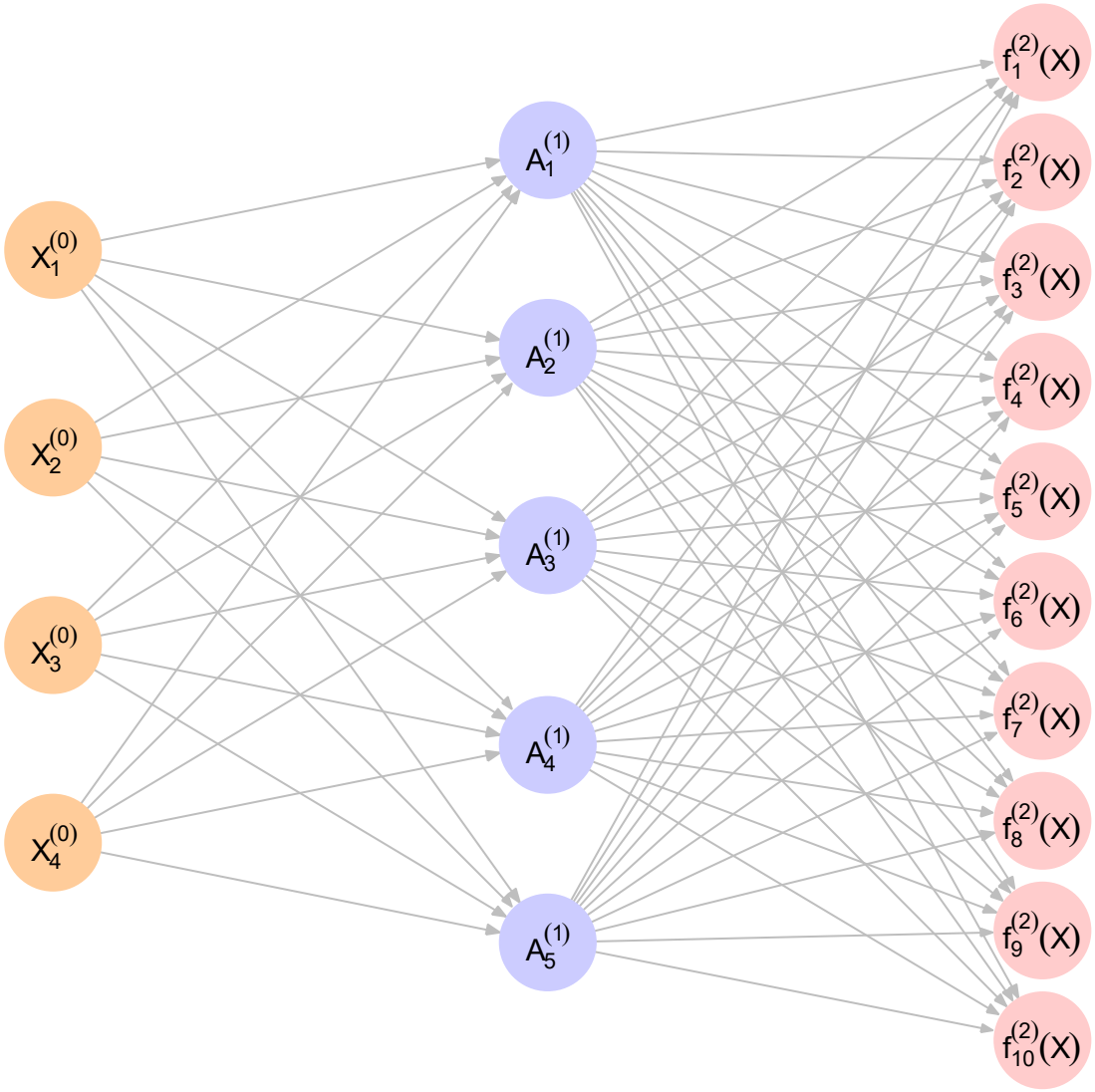
Joint presence-abundance model for a single species

g = sigmoid
Y1 = prob presence

g = identity
Y2 = abundance

Output is a prediction for two different outcome variables Y1, Y2

Architecture: multifunction (examples)



Could be quite complex with different g(z) for many different output variables

Deep learning

- Multilayer neural networks
  - aka deep feedforward networks
  - aka multilayer perceptrons (MLP)
- Model algorithm
  - expressiveness
    - ability to approximate complex nonlinearity
  - architecture: width versus depth

Expressiveness

Universal approximation theorem (1989)

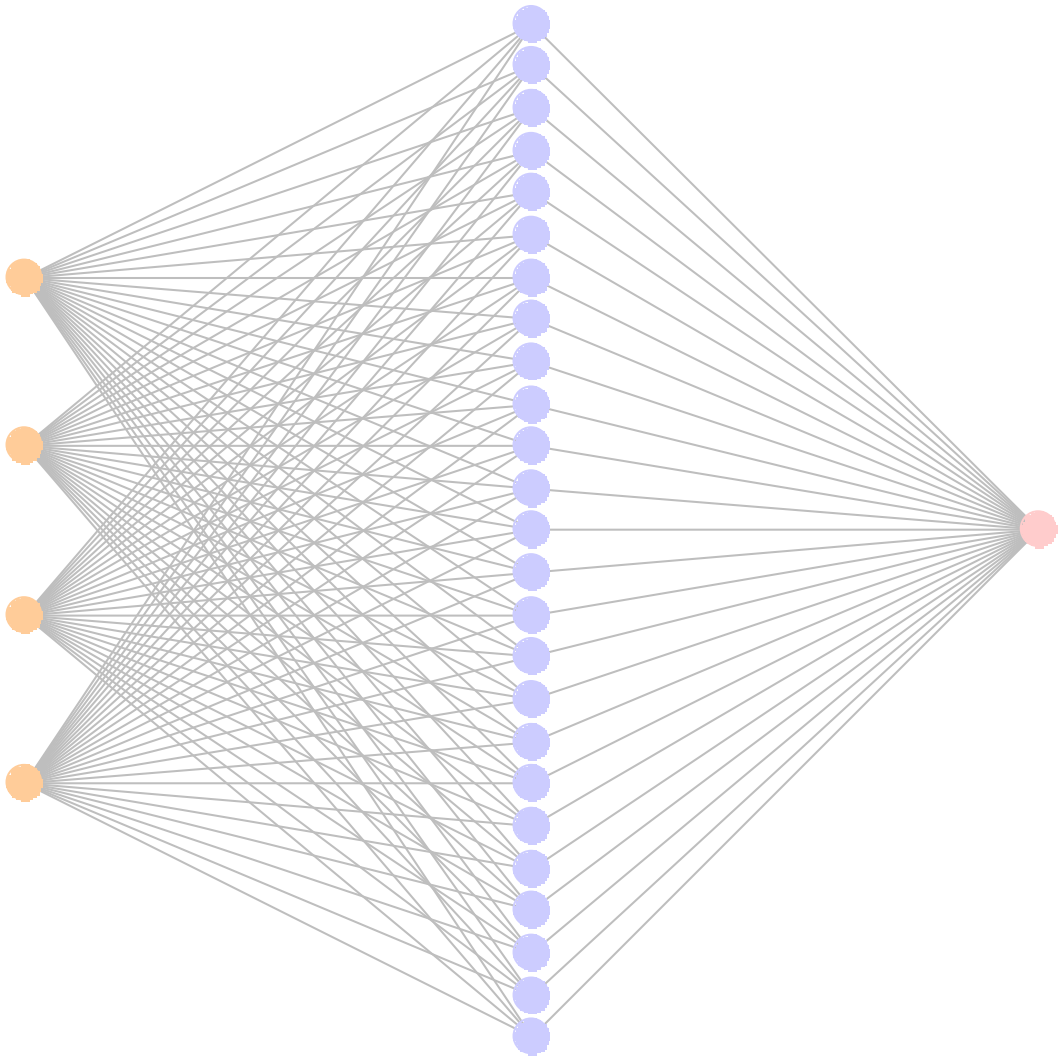
A feedforward neural network with at least one hidden layer can approximate any mapping

$$X \rightarrow f(X)$$

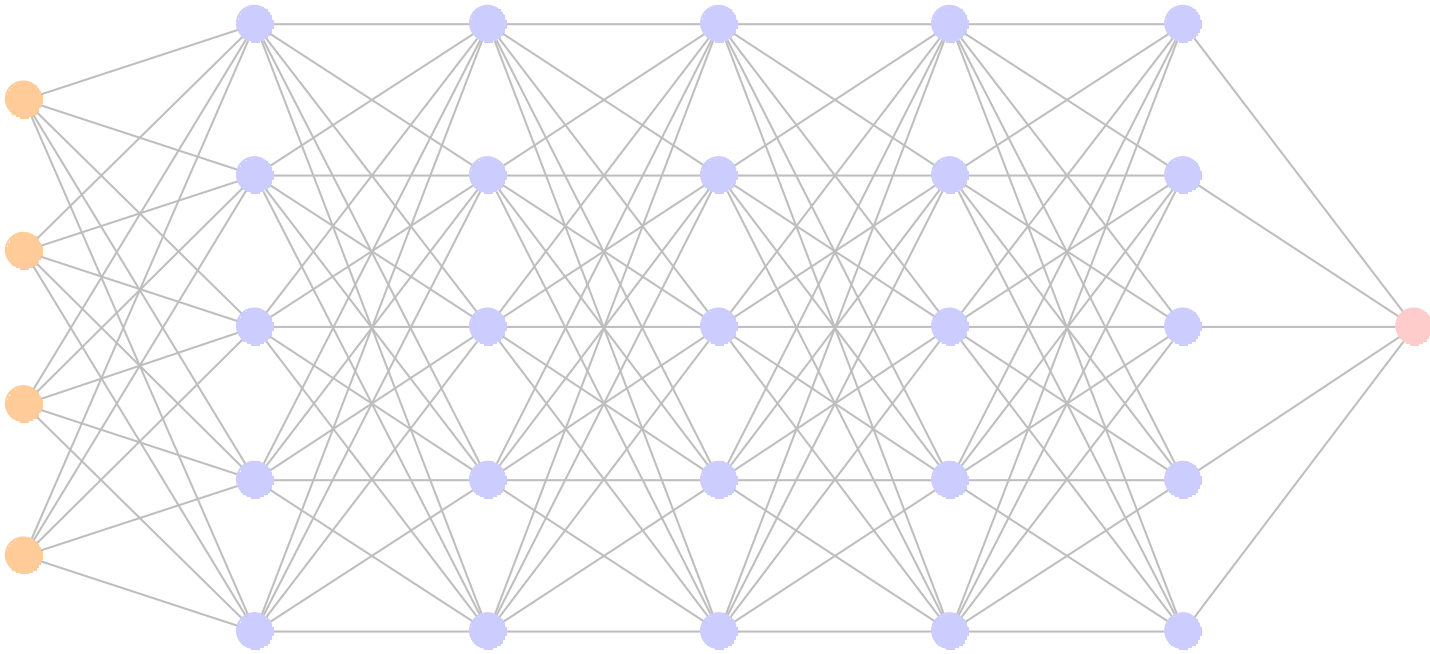
with arbitrarily low error provided it has enough units

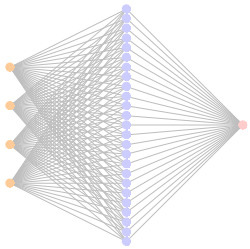
Can be wide or deep

Wide: 25 hidden units

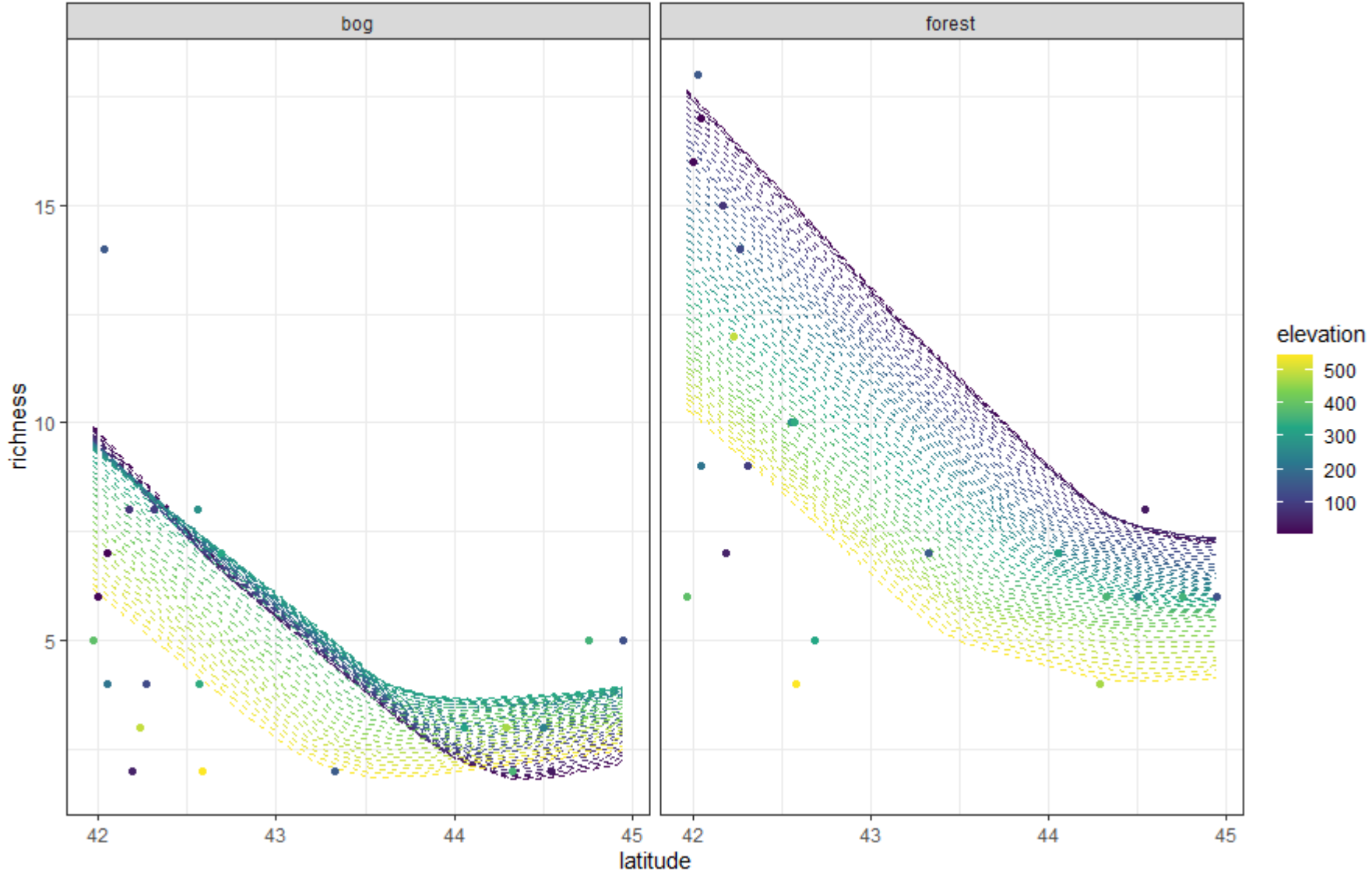


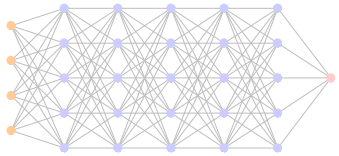
Deep: 25 hidden units



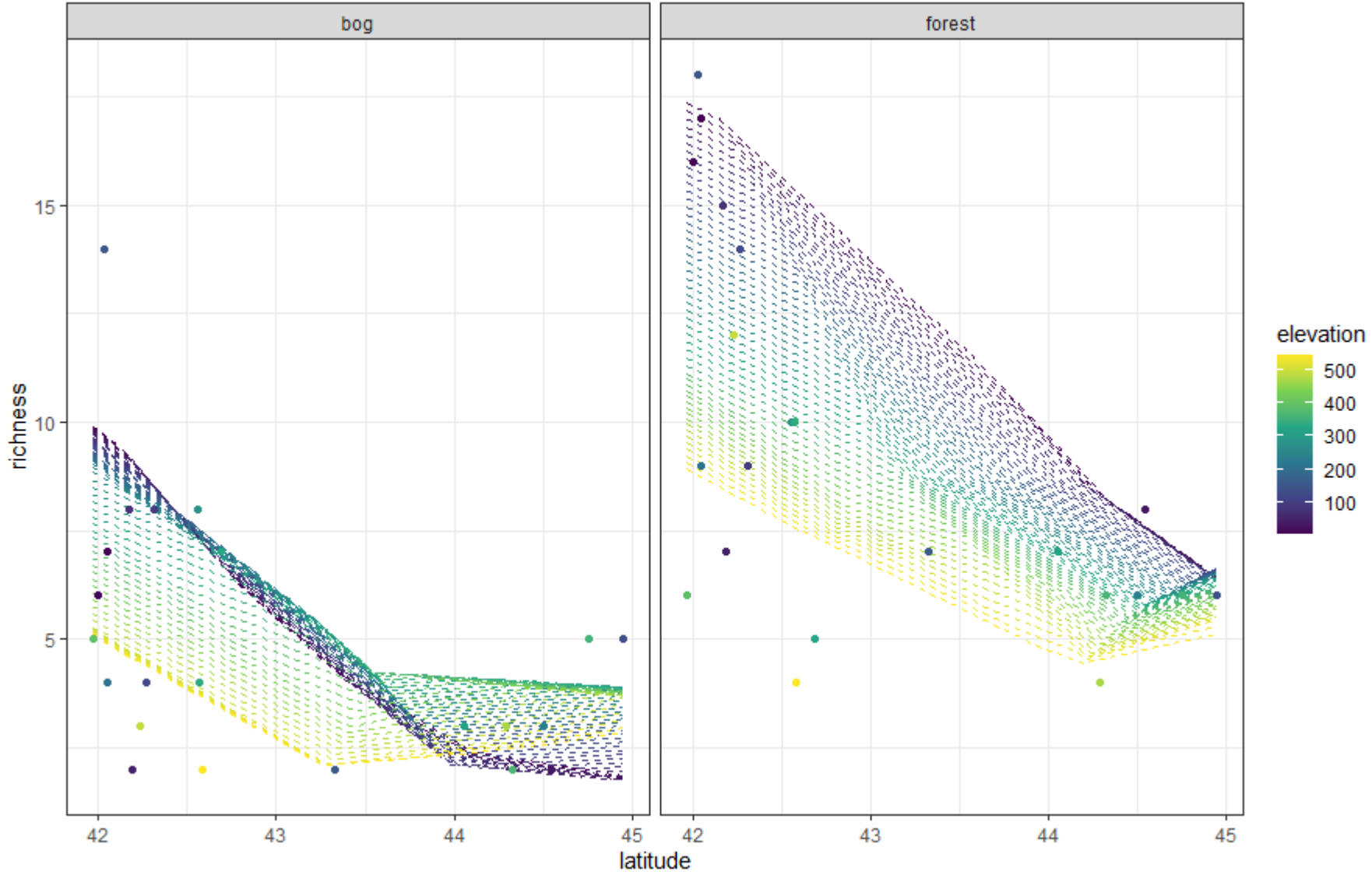


Ants data: wide



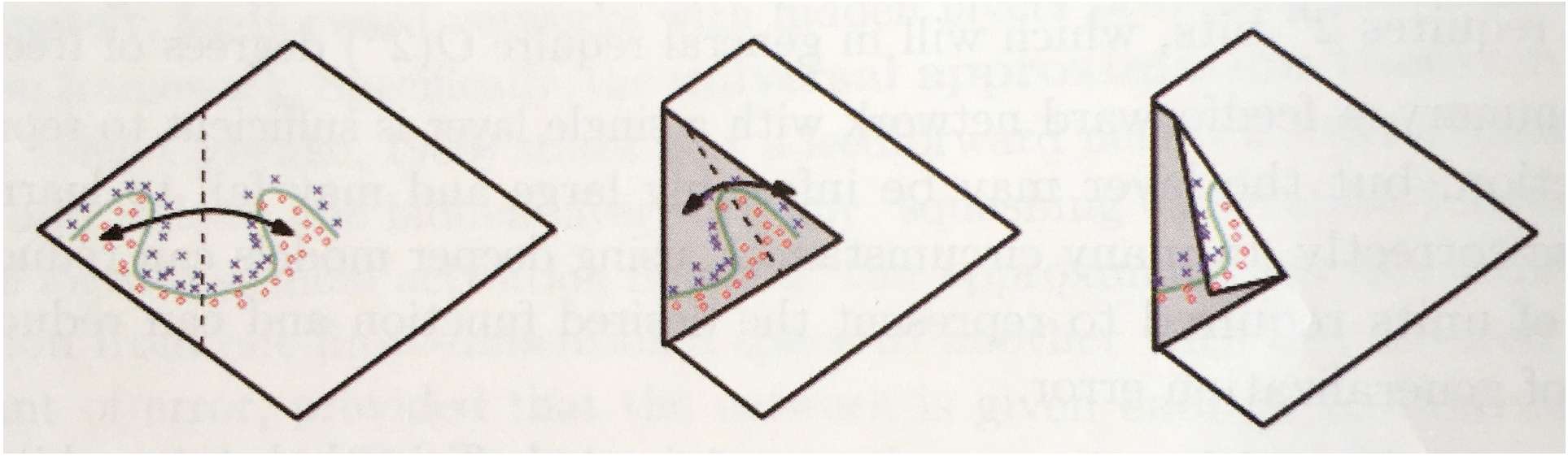


Ants data: deep



Deep: expressiveness

1 layer **2 layers** **3 layers**



Deeper networks (more layers) allow more folds, which can represent complex patterns more efficiently by finding symmetries

Goodfellow et al 2016 Deep Learning