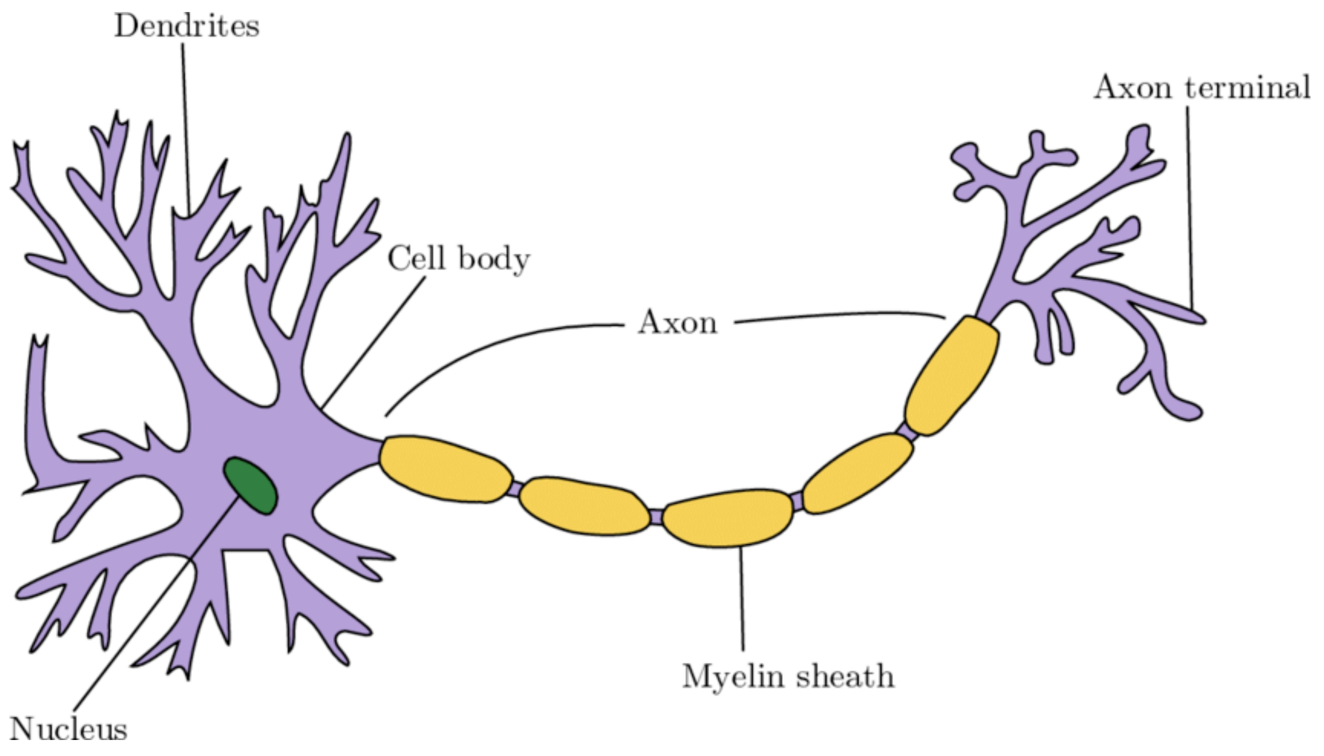
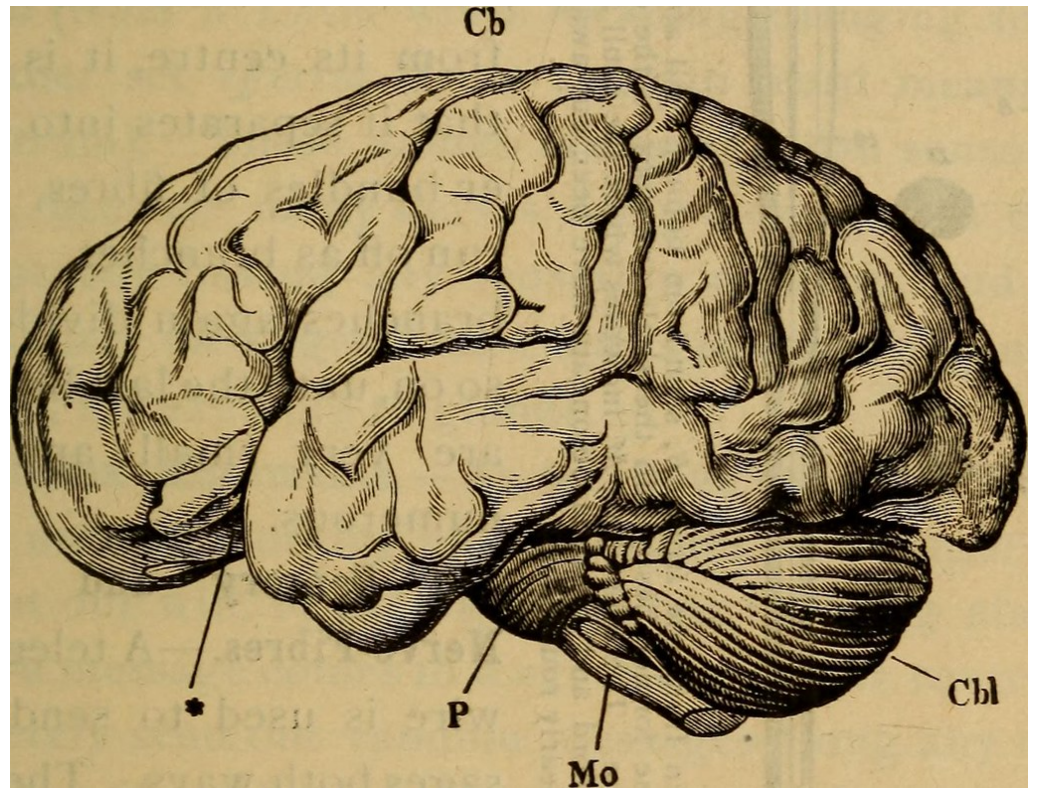
Neural Network Model

Neural Network Model

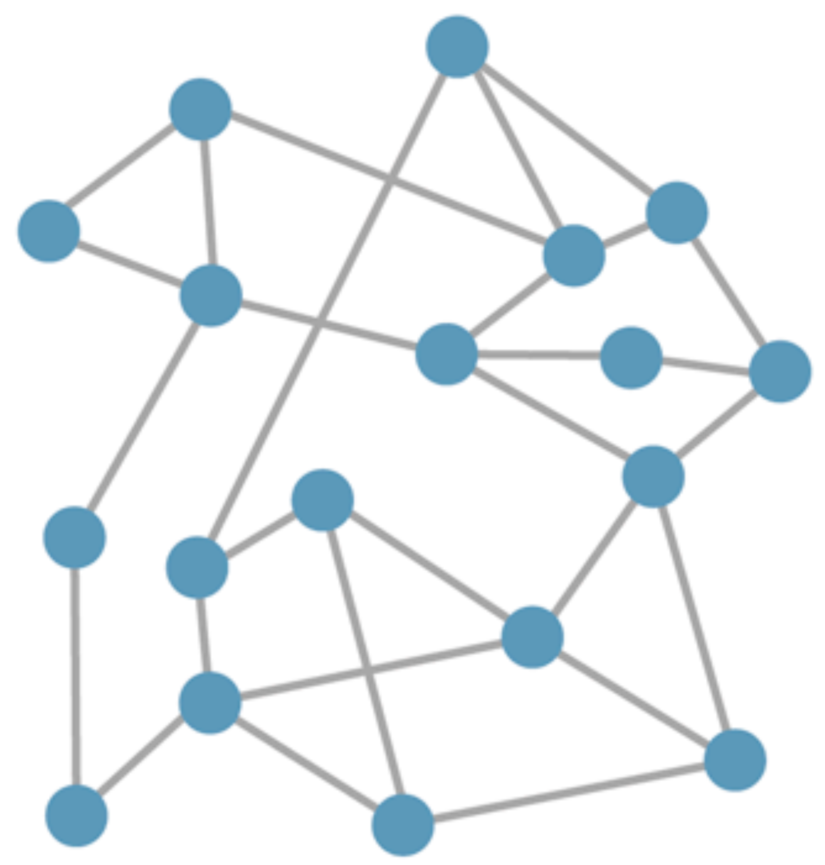




Neural Network Model

Distributed architecture
Connectionist basis

Neural Network Model



What is a model?

Neural Network Model

Useful representation

Often but not always an abstraction

~~Neural~~ Network Model

Models

For any word from vocabulary,
with a single guess

1. predict its frequency

2. predict the next word

Loss/cost function

$\lambda(y, \hat{y}) = (y - \hat{y})^2$ quadratic loss

$\lambda(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})$ hinge loss

$\log p(y \mid \theta)$ log predictive density

Loss/cost function

$\lambda(y, \hat{y}) = (y - \hat{y})^2$ quadratic loss

$\lambda(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})$ hinge loss

$\log p(y \mid \theta)$ log predictive density

To be a good model, you need to know what you will be evaluated on!

Loss/cost function

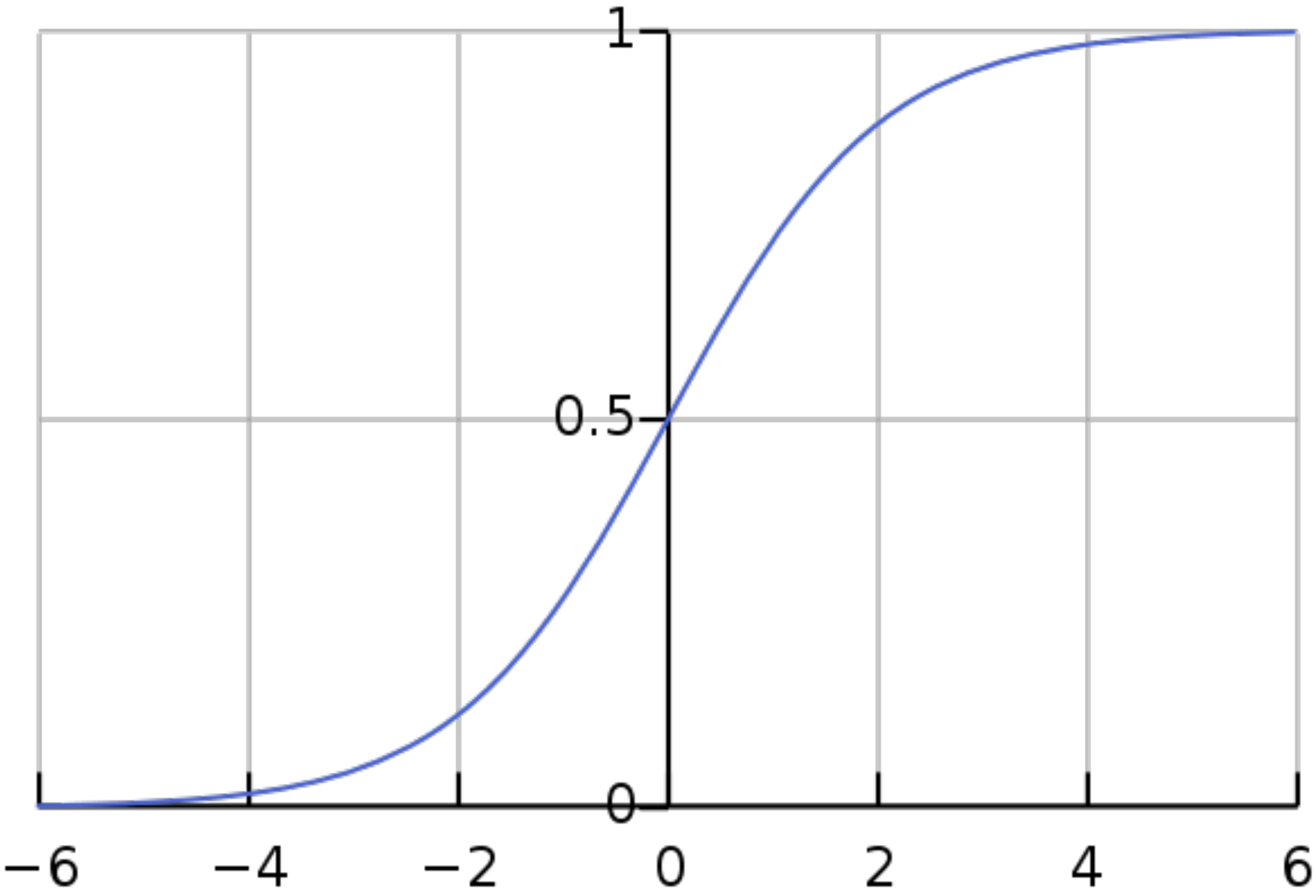
$\lambda(y, \hat{y}) = (y - \hat{y})^2$ quadratic loss

$\lambda(y, \hat{y}) = \max(0, 1 - y \cdot \hat{y})$ hinge loss

$\log p(y \mid \theta)$ log predictive density

There is no free lunch

Linear models and non-linearity



Non-linearity, at last!

Generalized linear models are still linear models even though they use a non-linear transformation

Generalized linear models are still linear models even though they use a non-linear transformation

They explicitly estimate the effect of one or more predictors on an outcome

Generalized linear models are still linear models even though they use a non-linear transformation

They explicitly estimate the effect of one or more predictors on an outcome

NNs scale up these ideas but are non-linear and have many parameters with no clear semantics behind them

Neural Network models

Non-linearity: Many phenomena are non-linear, so linearity is a potentially unnecessary constraint in the relationship between input and output

Parameters with no clear semantics: Automatically induced from data with no need to match architecture to phenomenon*

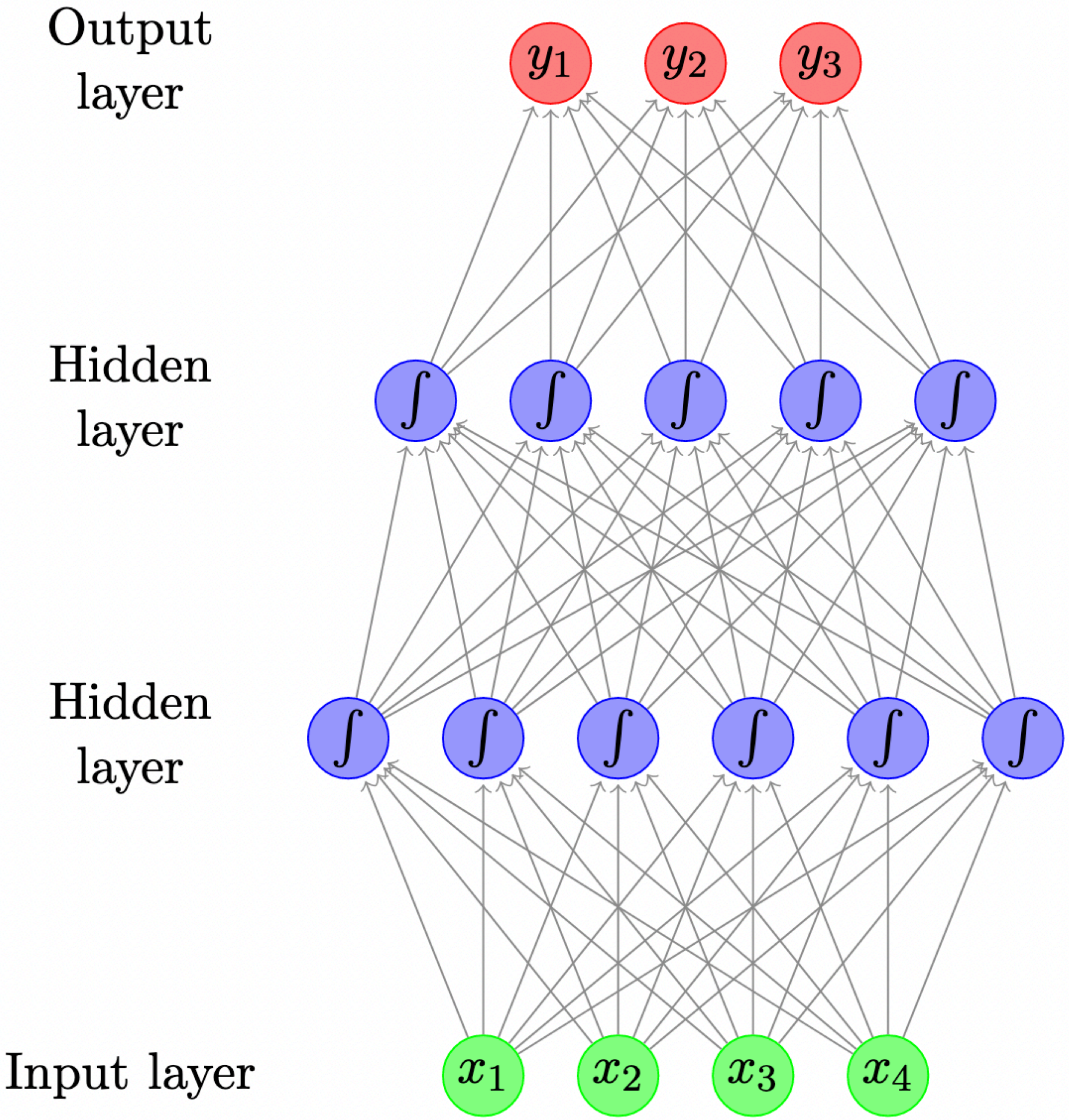
Many parameters: Can be an issue but doesn’t need to be

Dense representations

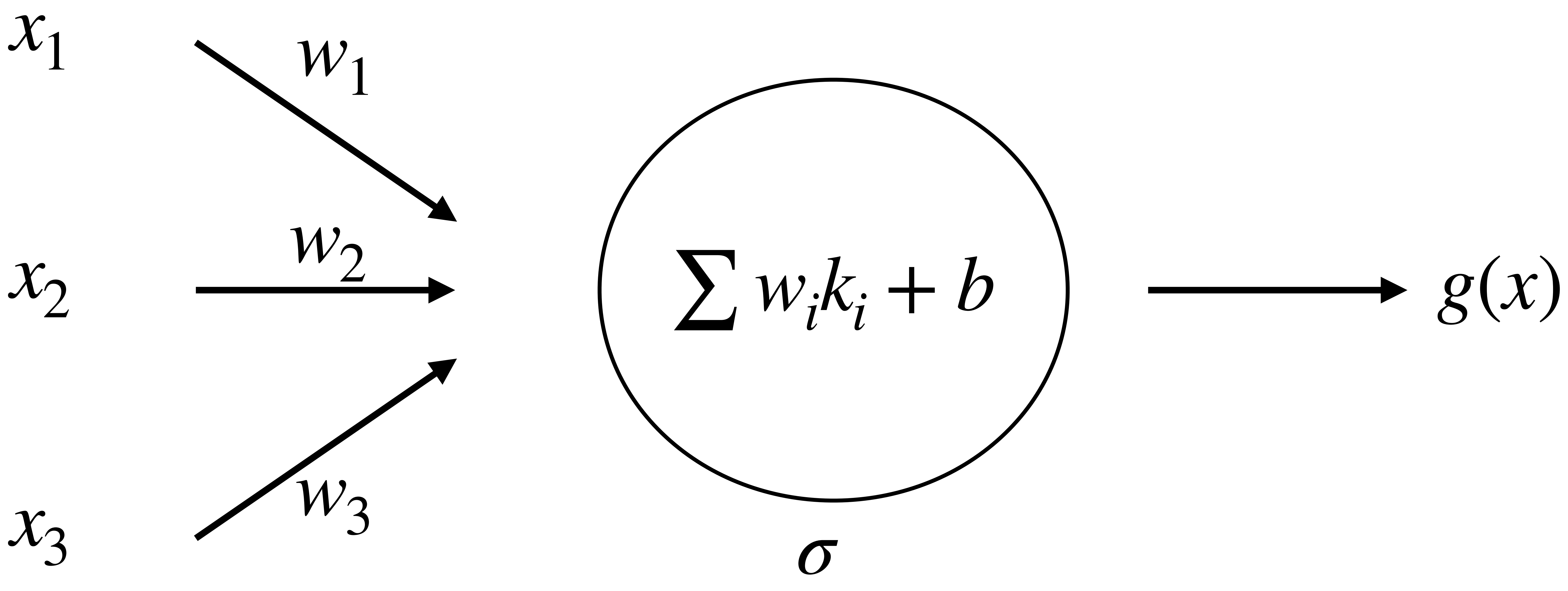
Feed-forward NN

$$[x_1, \ldots, x_{d_{in}}] \quad \Rightarrow \quad \text{NN}(\,\cdot\,) \quad \Rightarrow [y_1, \ldots, y_{d_{out}}]$$

Feed-forward NN

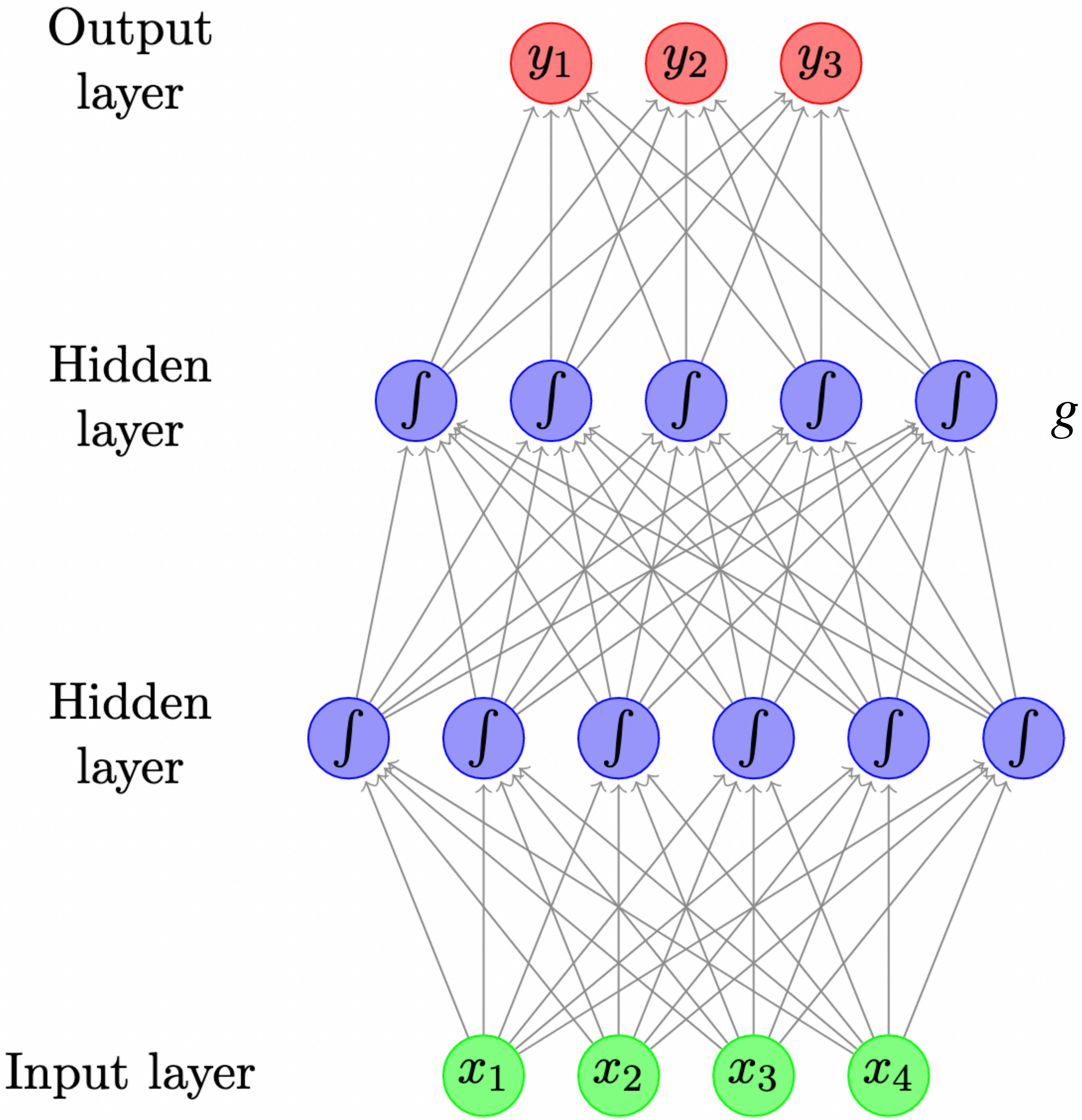


Feed-forward NN



$$\sigma(\sum_1^3 w_i k_i + b) = g(x)$$

Feed-forward NN



$g^2(g^1(xW^1 + b^1)W^2 + b^2)$

$g^1(xW^1 + b^1)$

$x \in \mathbb{R}^{d_{in}}$

Decisions for the modeller/engineer

1. Number of dimensions

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

3. Non-linearities (e.g., sigmoid, tanh, rectifier)

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

3. Non-linearities (e.g., sigmoid, tanh, rectifier)

4. Output transformation (e.g. softmax)

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

3. Non-linearities (e.g., sigmoid, tanh, rectifier)

4. Output transformation (e.g. softmax)

5. Connectivity

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

3. Non-linearities (e.g., sigmoid, tanh, rectifier)

4. Output transformation (e.g. softmax)

5. Connectivity

6. Loss function

Decisions for the modeller/engineer

1. Number of dimensions

2. Number of layers

3. Non-linearities (e.g., sigmoid, tanh, rectifier)

4. Output transformation (e.g. softmax)

5. Connectivity

6. Loss function

7. Training regime
   (e.g., stochastic gradient descent + flavor;
   batching; drop-out)