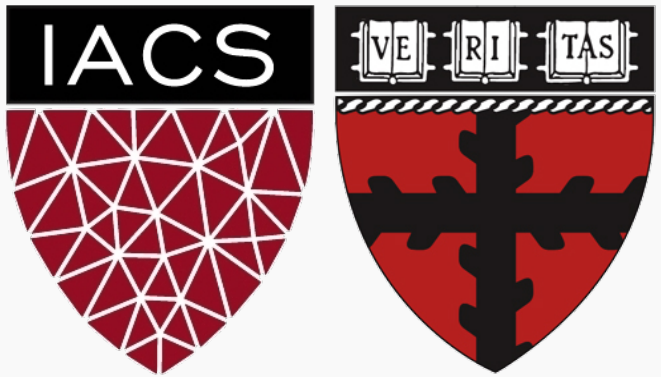
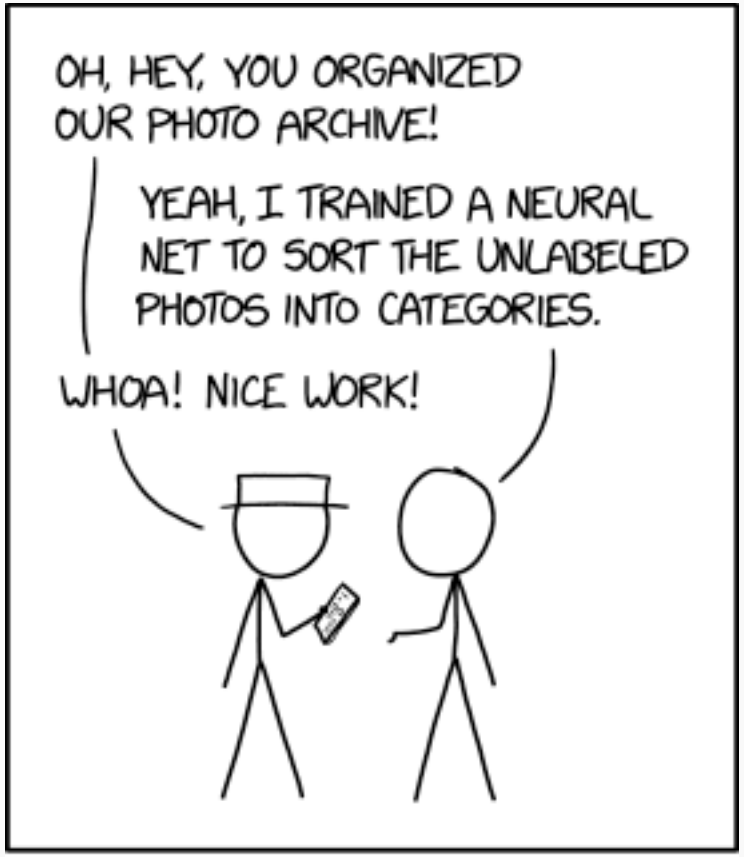
Anatomy of NN

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Tanner





ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

Outline

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Outline

Anatomy of a NN

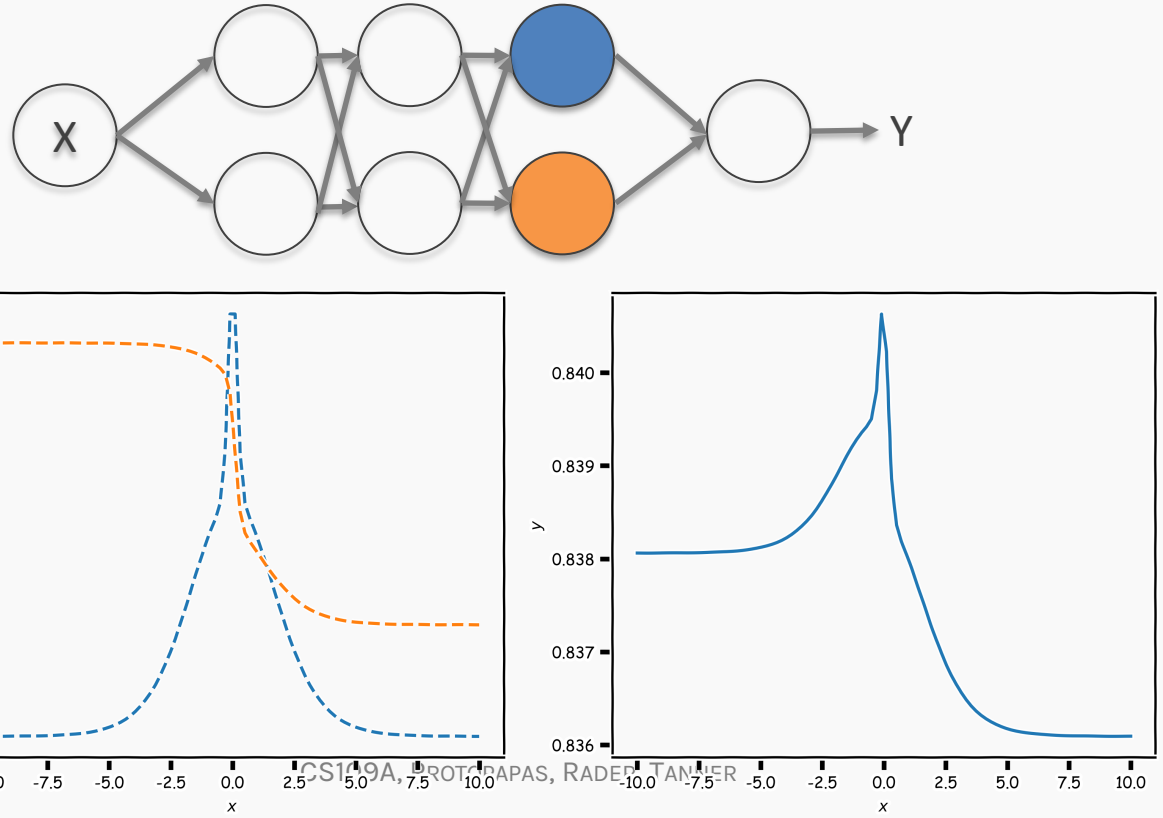
Design choices

- Activation function
- Loss function
- Output units
- Architecture

The central idea behind Neural Networks

Building a model that is both complex and simple to manipulate by **composing** multiple functions together.

Example:



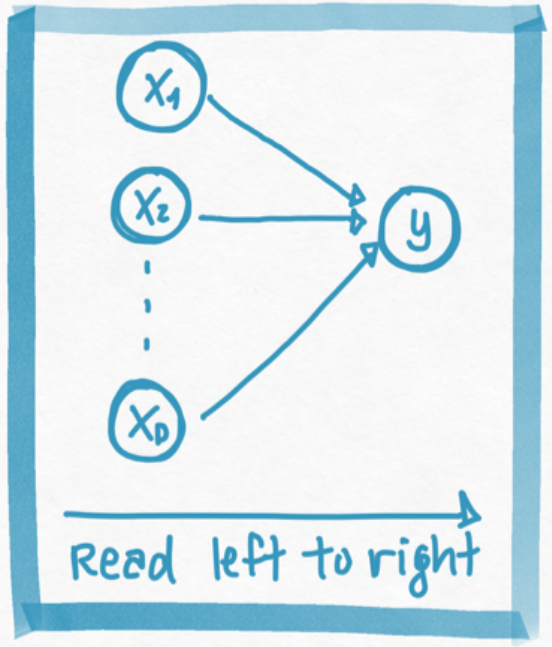
Graphical representation of simple functions

We build these complex functions by composing simple functions of the form:

$$h_w(x) = f(XW + b)$$

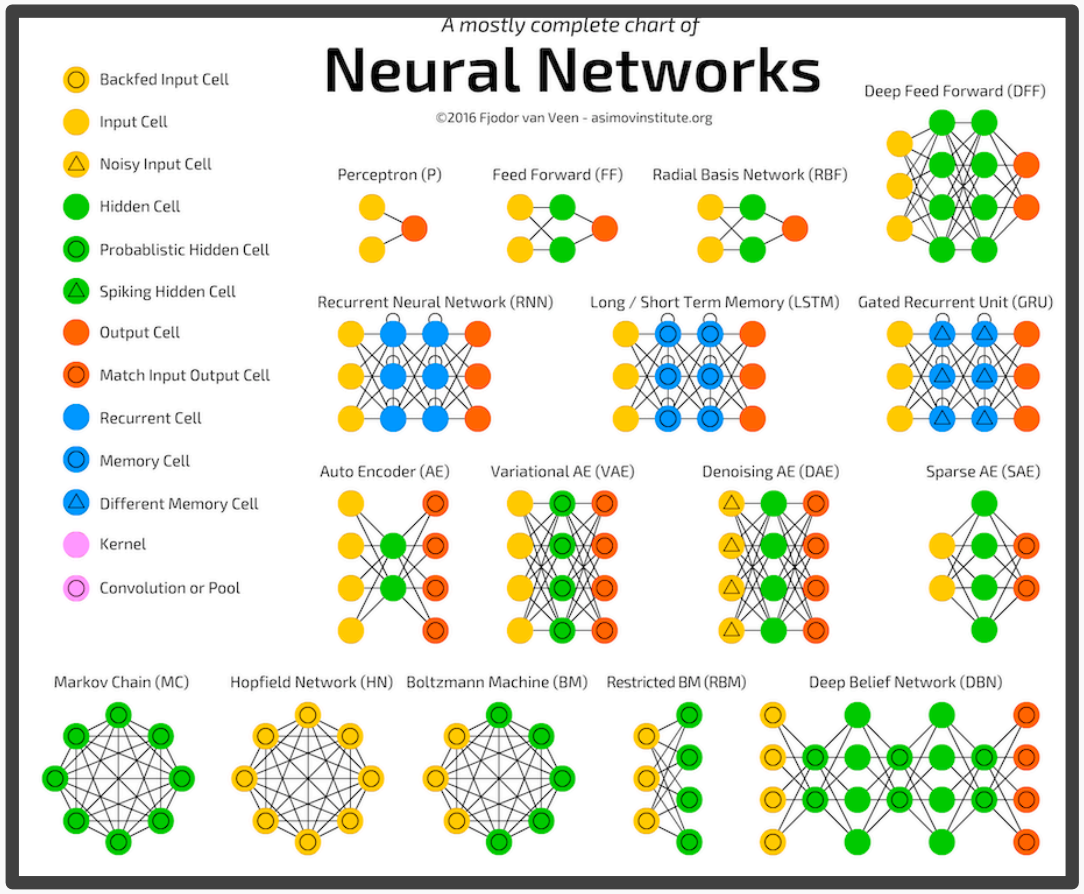
where f is the activation function.

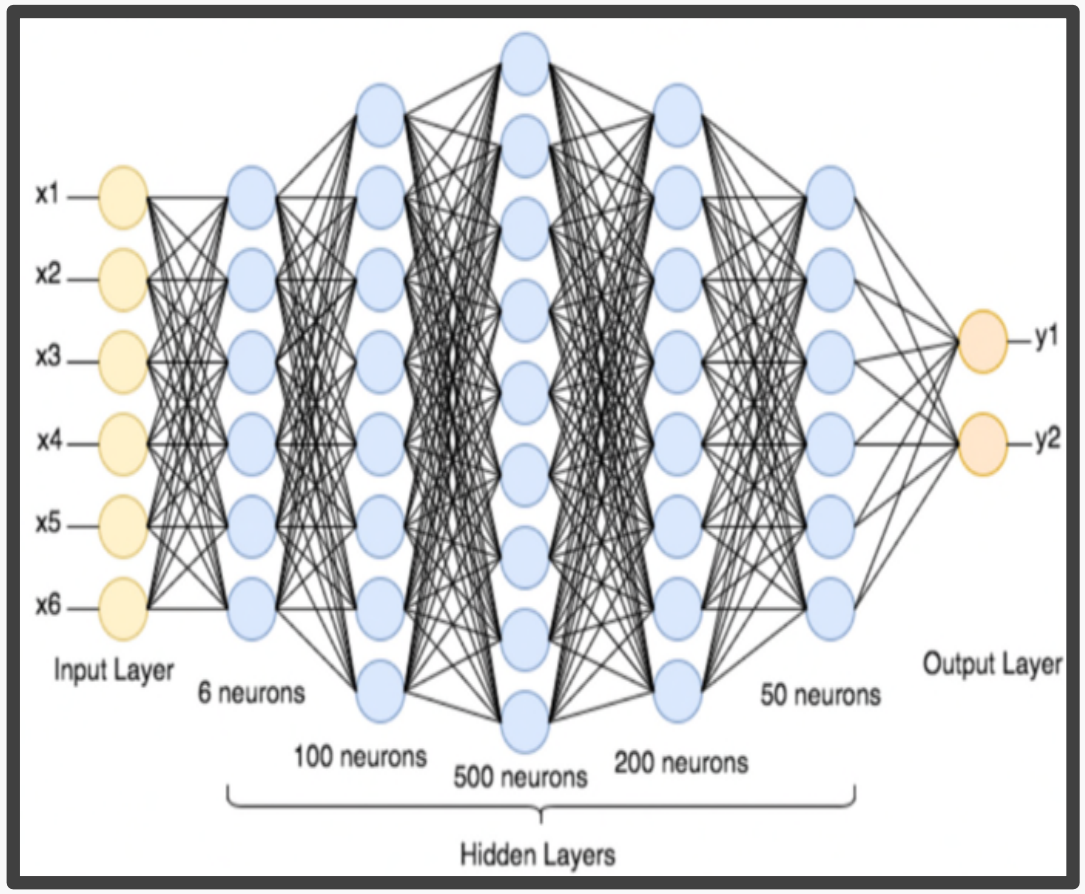
We represent our simple function as a **graph**



Each edge in this graph represents multiplication by a different weight, w_i.

The zoo of neural network architectures

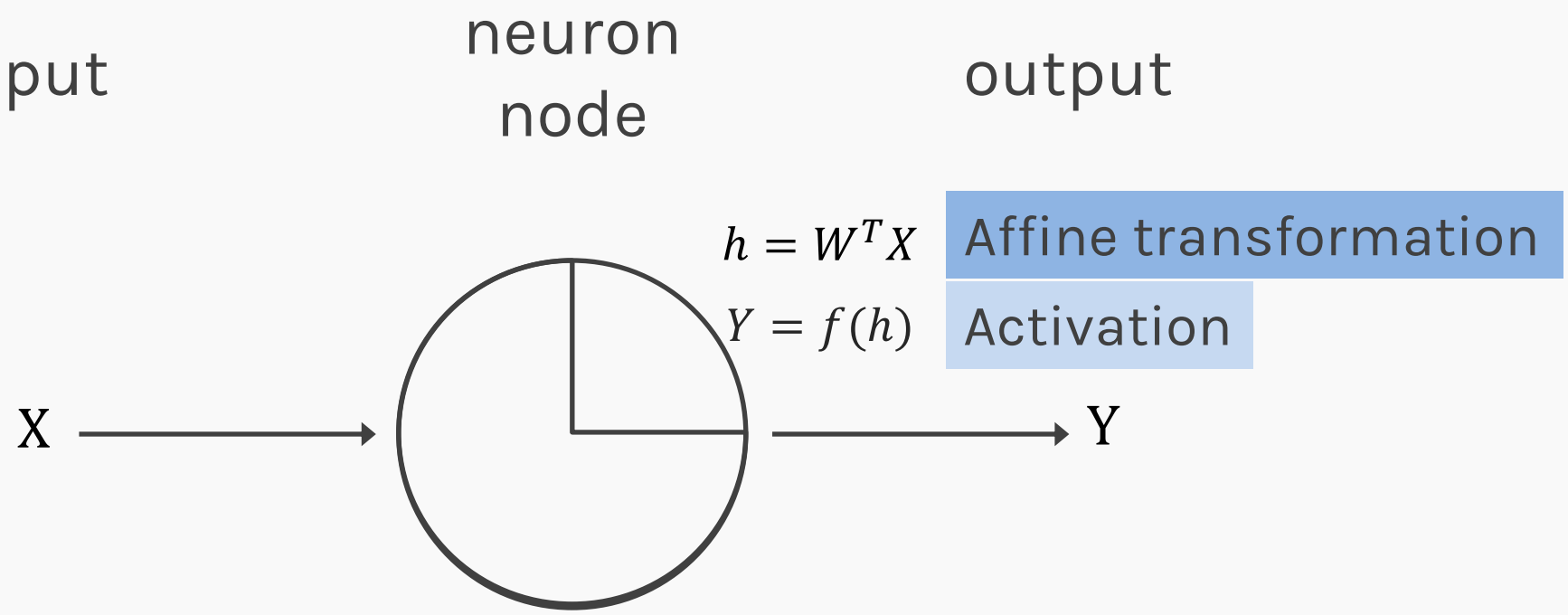




Different architectures result into functions with very different properties.

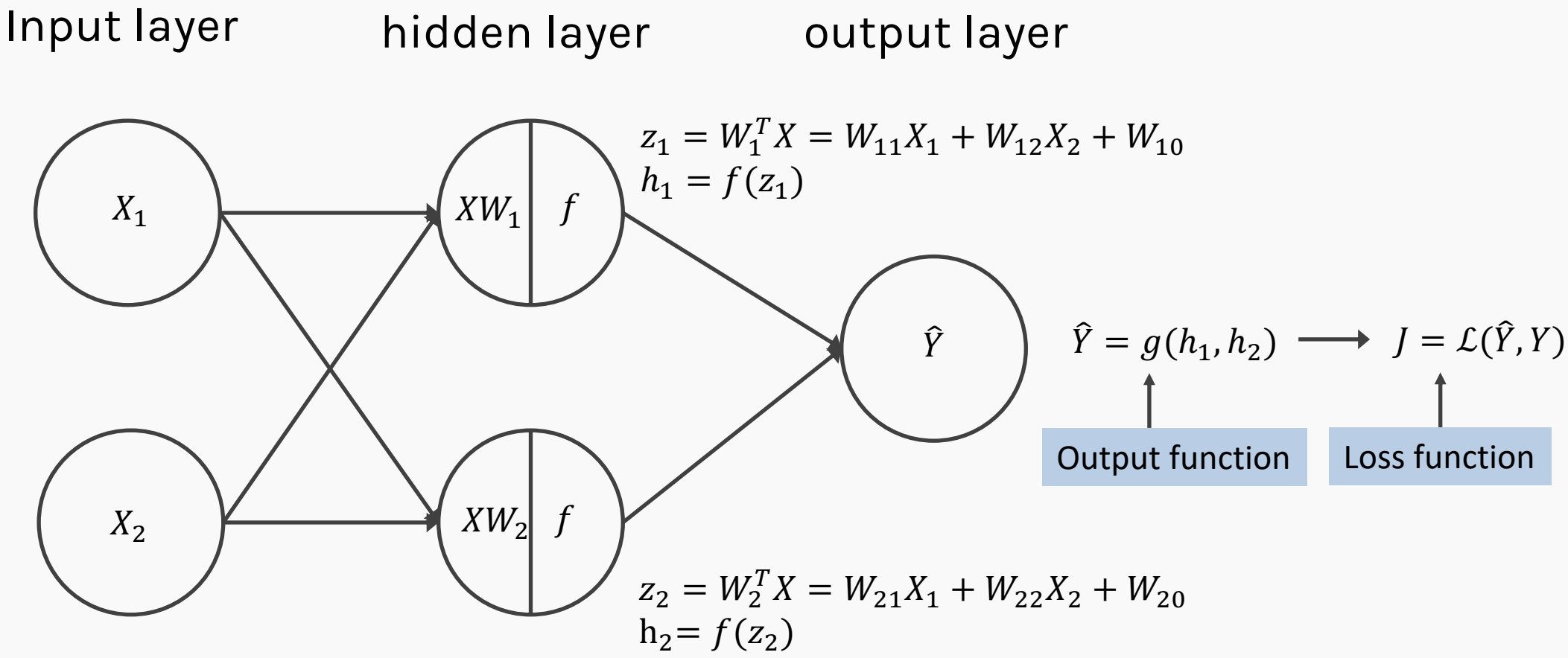
Larger networks can express more complex functions

Anatomy of artificial neural network (ANN)



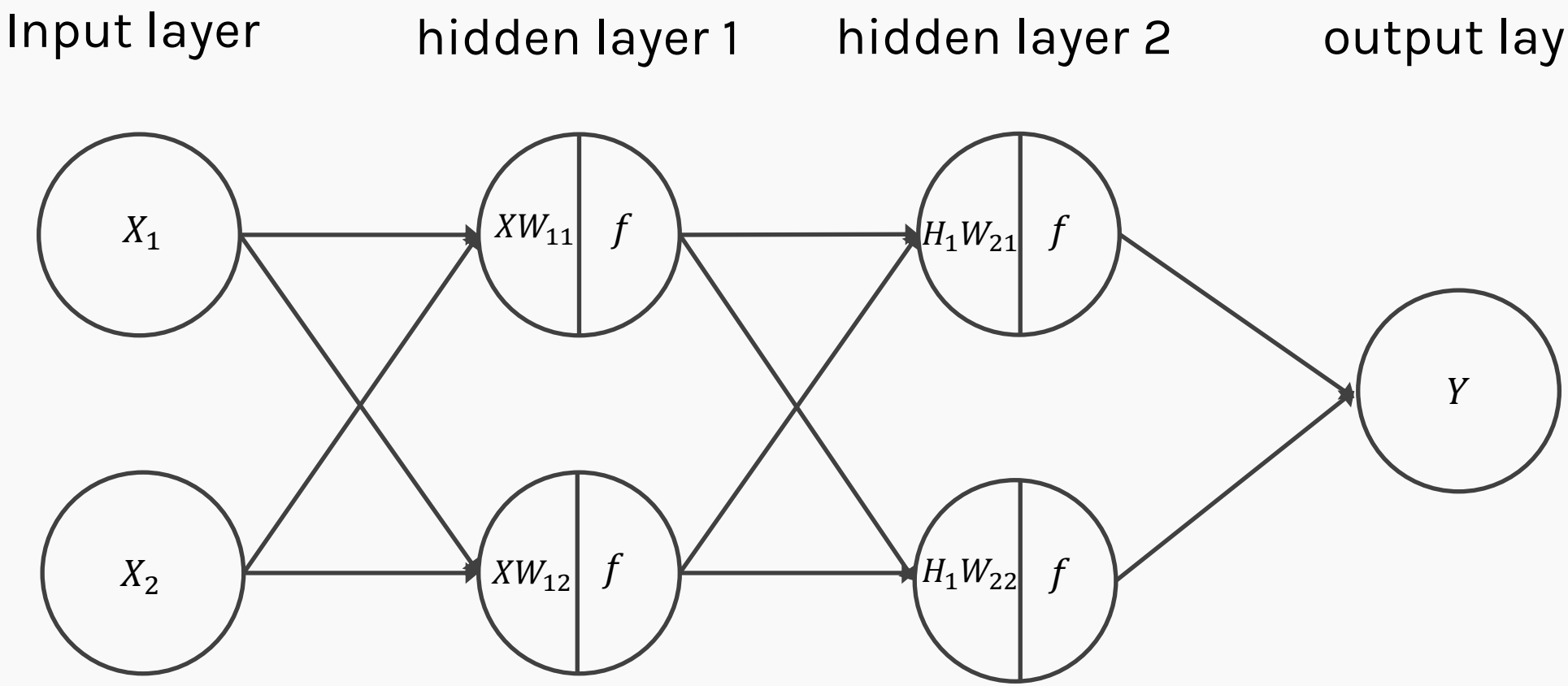
We will talk later about the choice of activation function. So far we have only talked about sigmoid as an activation function but there are other choices.

Anatomy of artificial neural network (ANN)

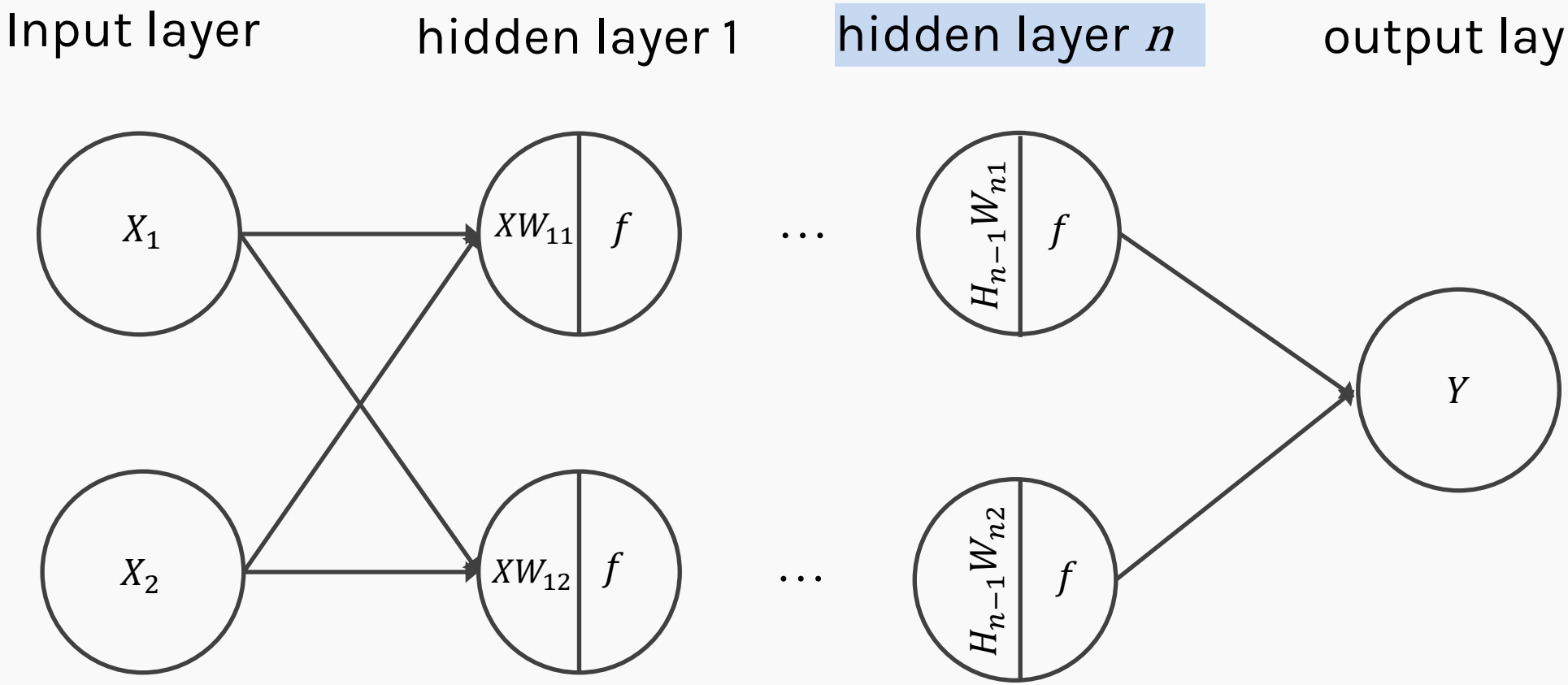


We will talk later about the choice of the output layer and the loss function. So far we consider sigmoid as the output and log-bernouli.

Anatomy of artificial neural network (ANN)

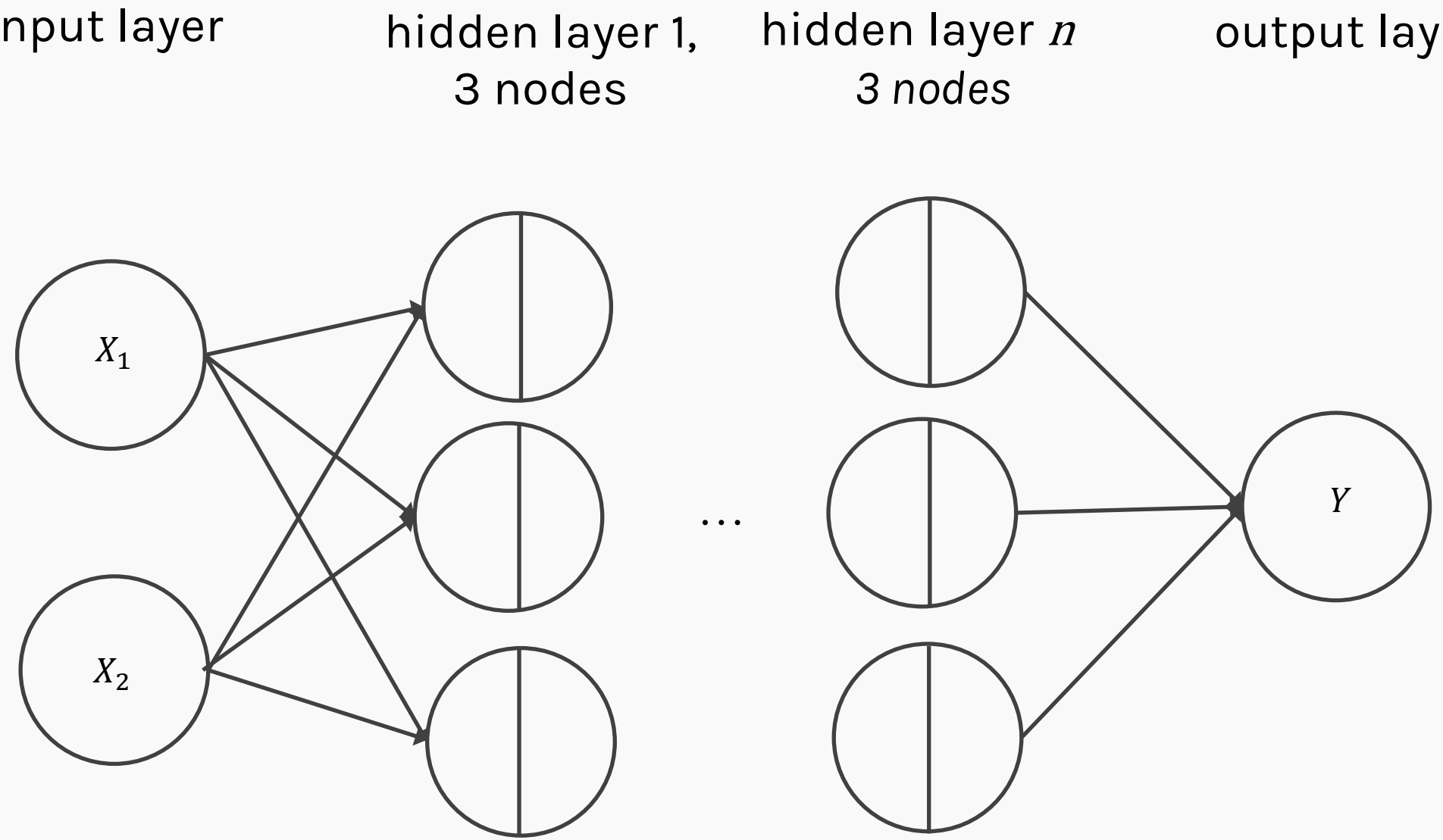


Anatomy of artificial neural network (ANN)



We will talk later about the choice of the number of layers.

Anatomy of artificial neural network (ANN)



Anatomy of artificial neural network (ANN)

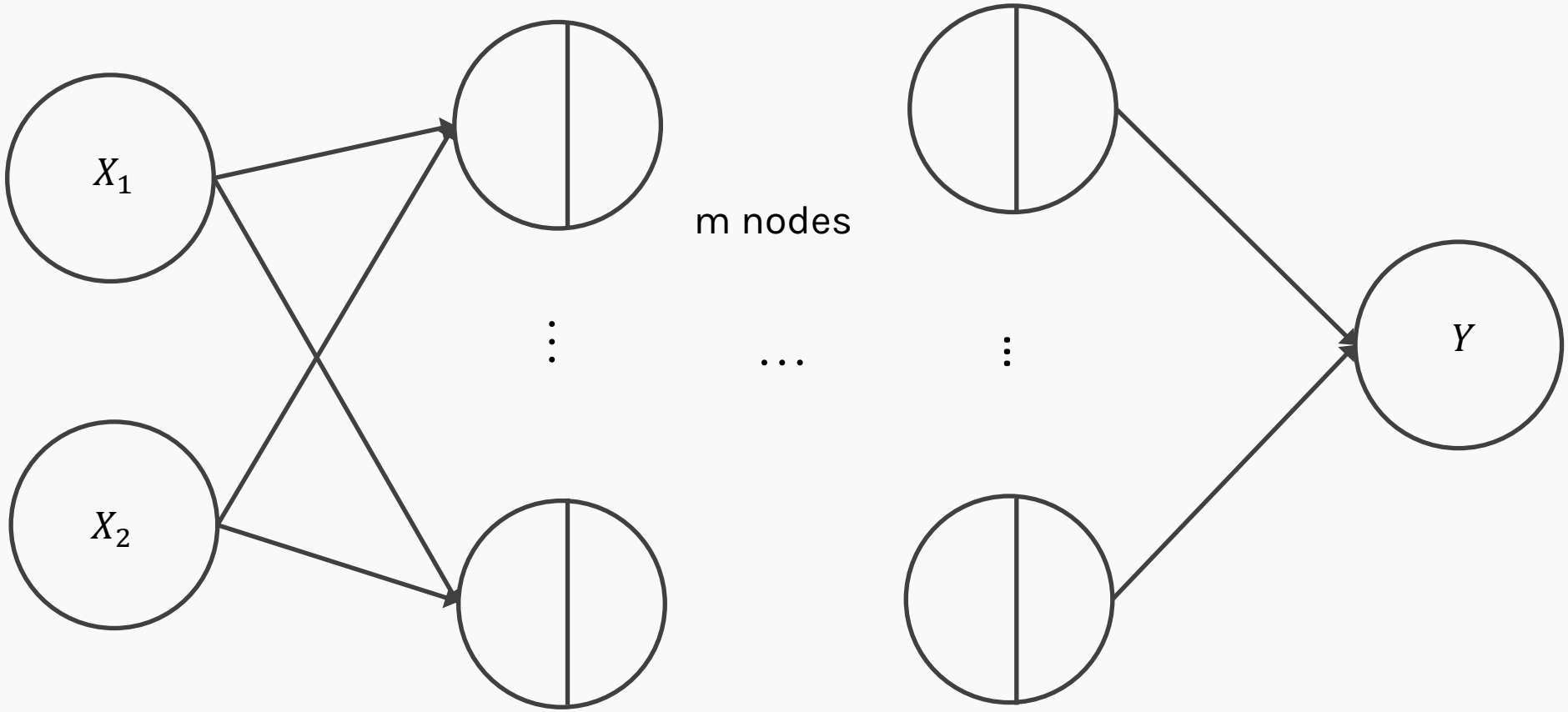
Input layer

hidden layer 1,

m nodes

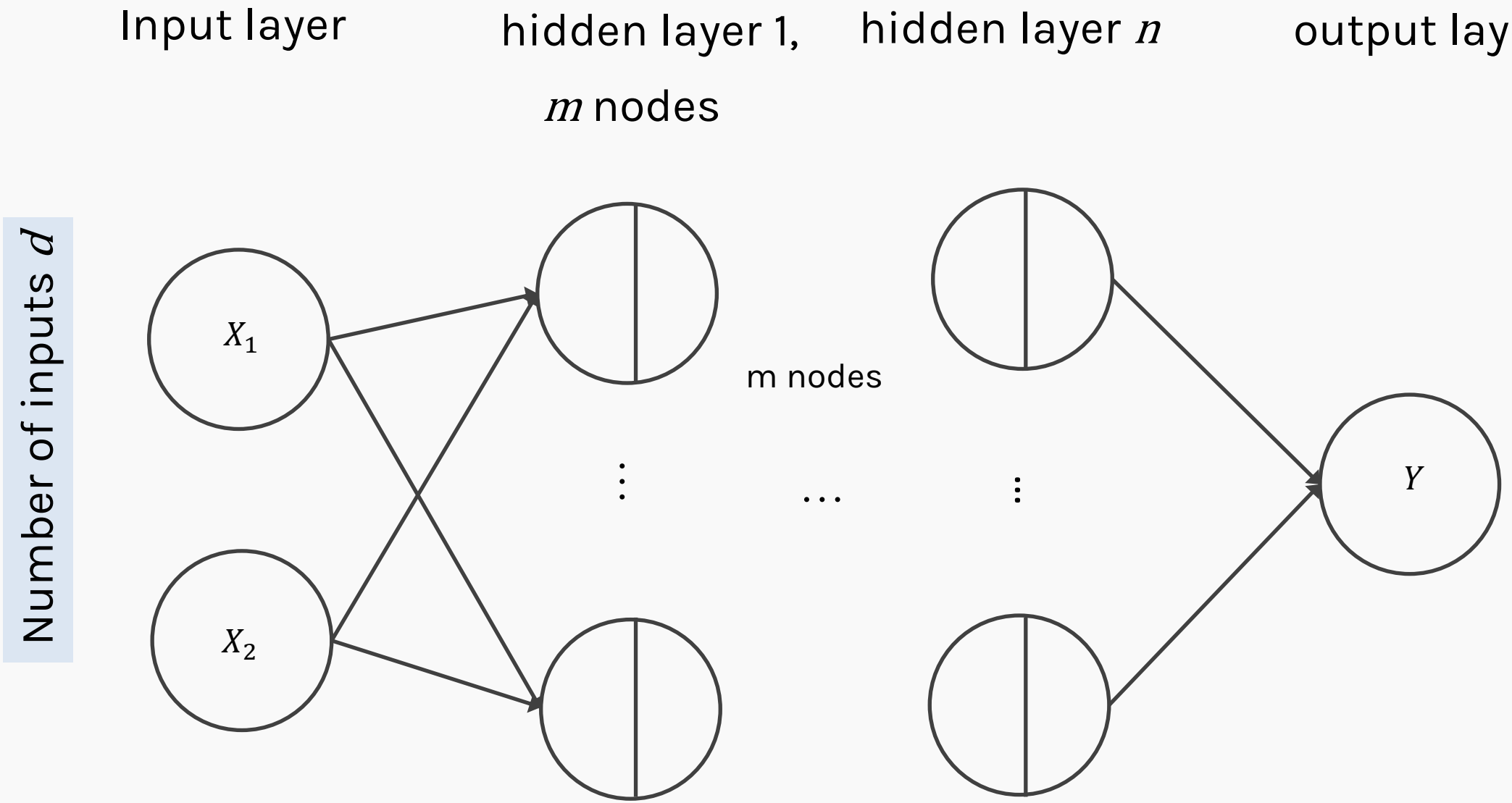
hidden layer n

output layer



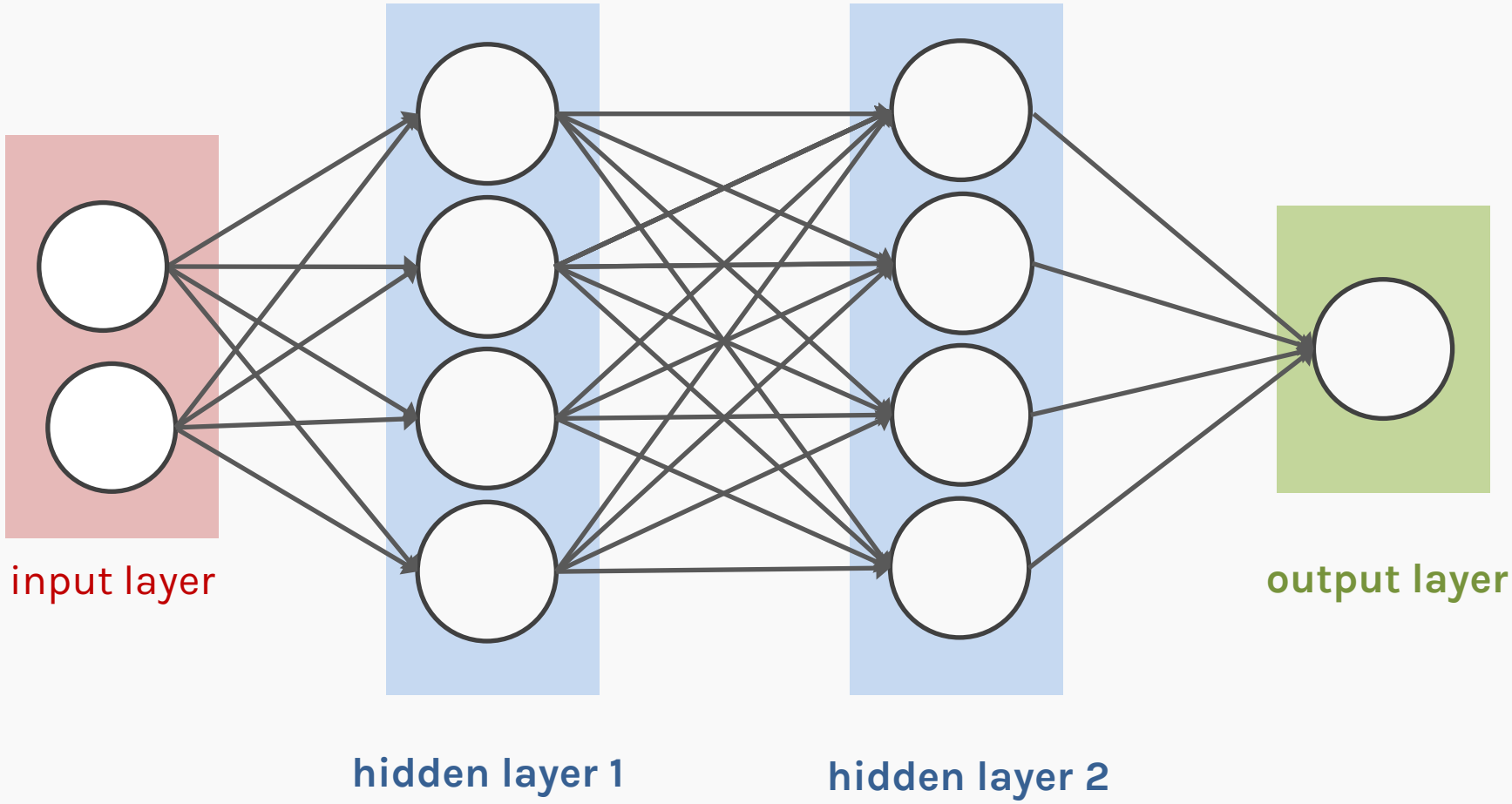
We will talk later about the choice of the number of nodes.

Anatomy of artificial neural network (ANN)

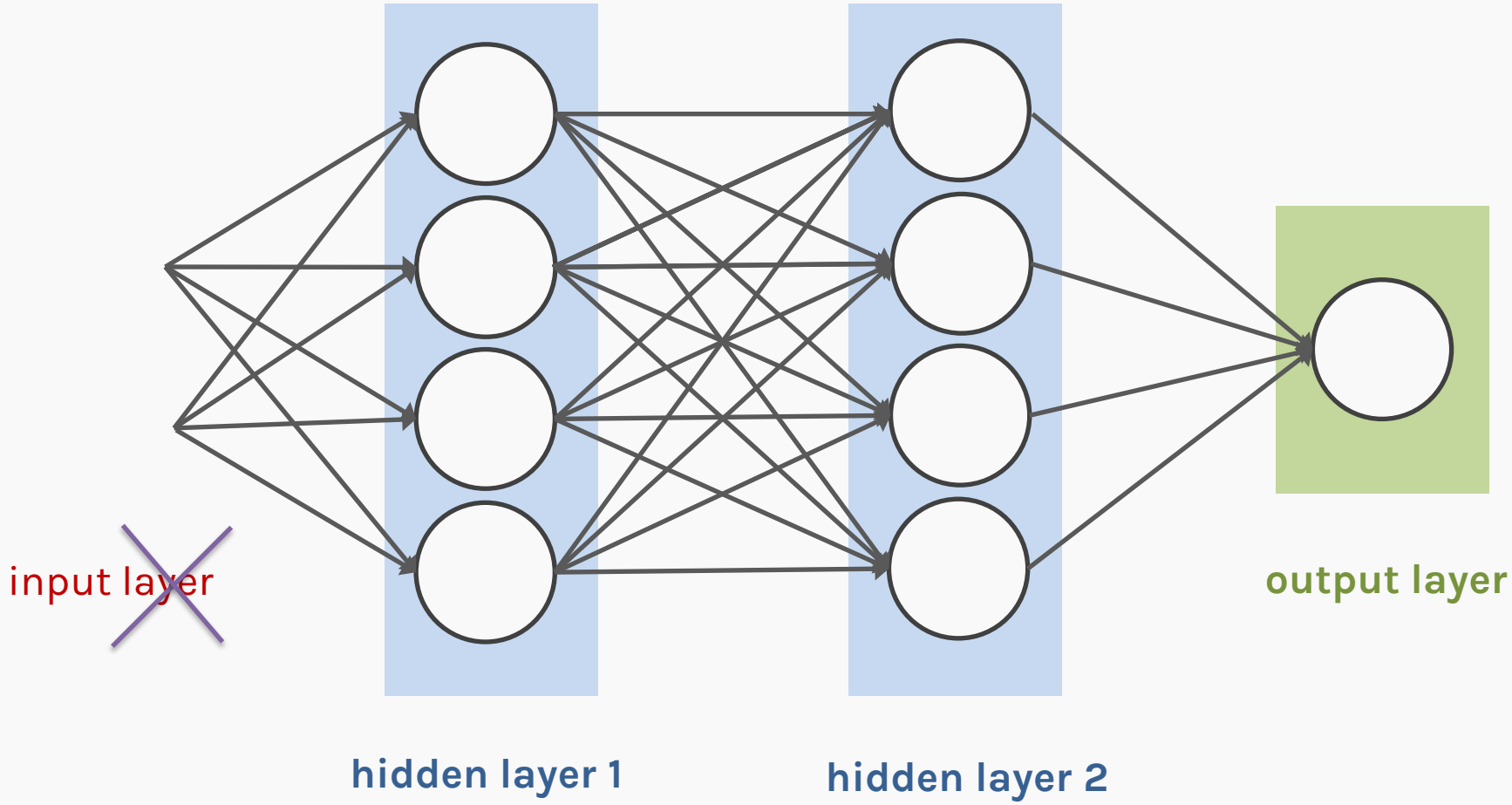


Number of inputs is specified by the data

Anatomy of artificial neural network (ANN)



Anatomy of artificial neural network (ANN)



Outline

Anatomy of a NN

Design choices

- Activation function
- Loss function
- Output units
- Architecture

Outline

Anatomy of a NN

Design choices

- **Activation function**
- Loss function
- Output units
- Architecture

Activation function

$$h = f(W^T X + b)$$

The activation function should:

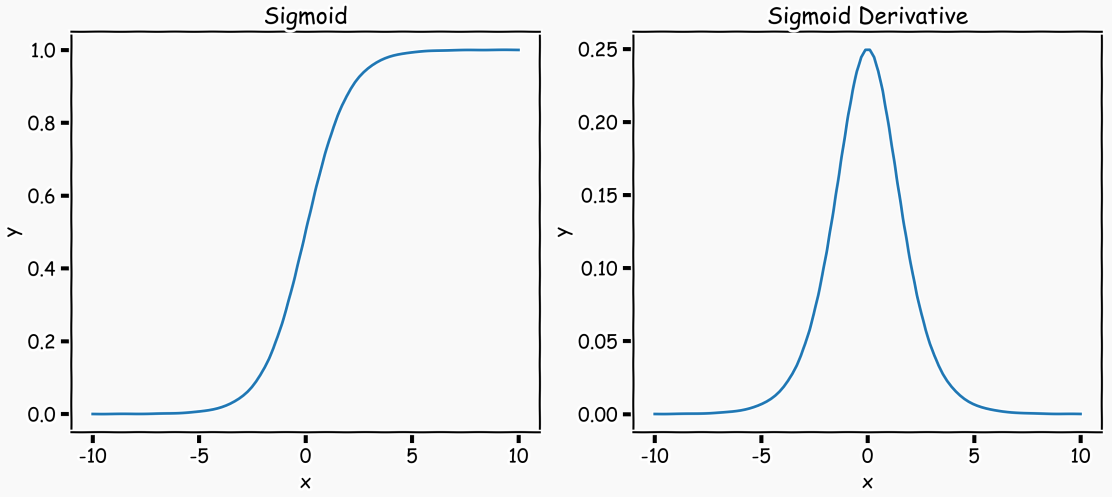
- Provide non-linearity
- Ensure gradients remain large through hidden unit

Common choices are

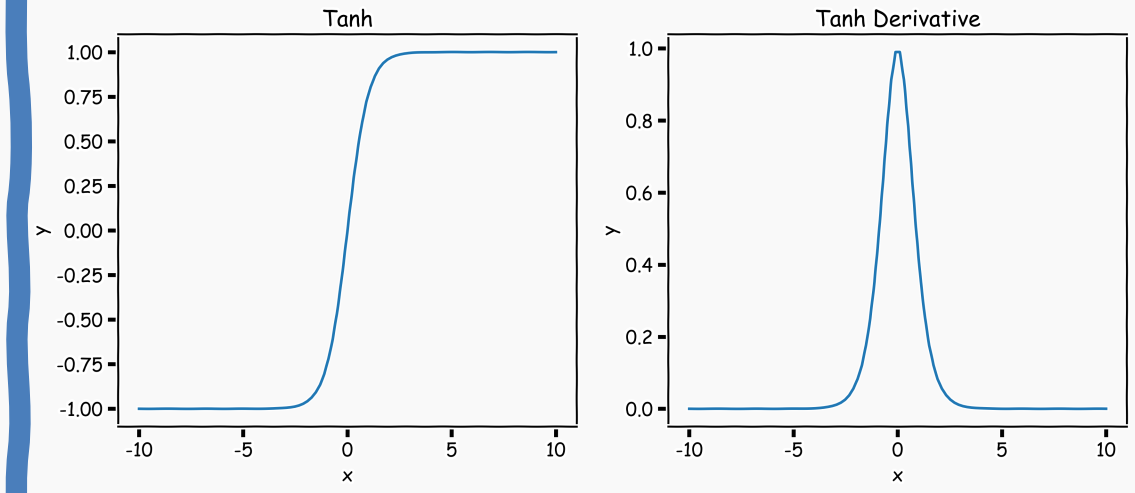
- sigmoid, tanh
- ReLU, leaky ReLU, Generalized ReLU
- softplus
- swish

Sigmoid, $\sigma()$ (aka logistic) and tanh

$$y = \frac{1}{1 + e^{-x}}$$



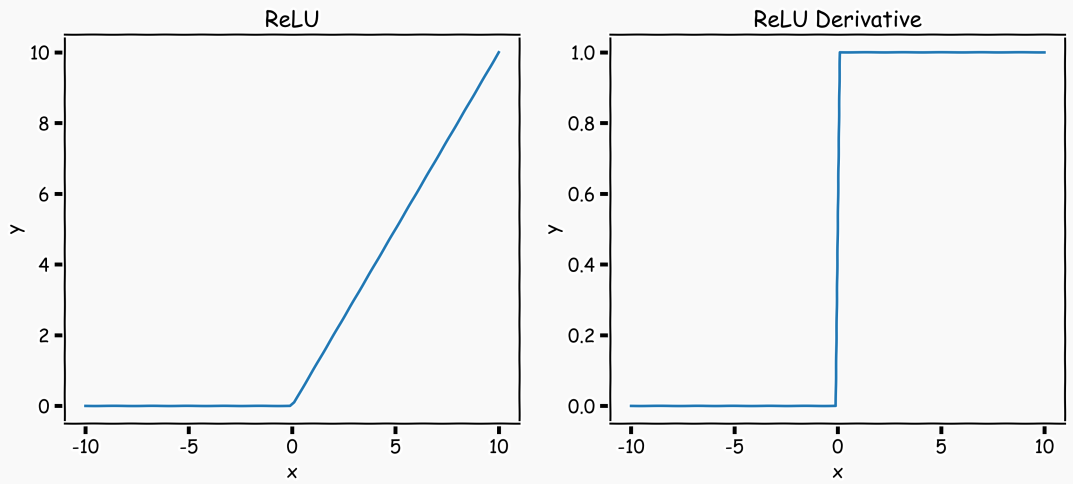
$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



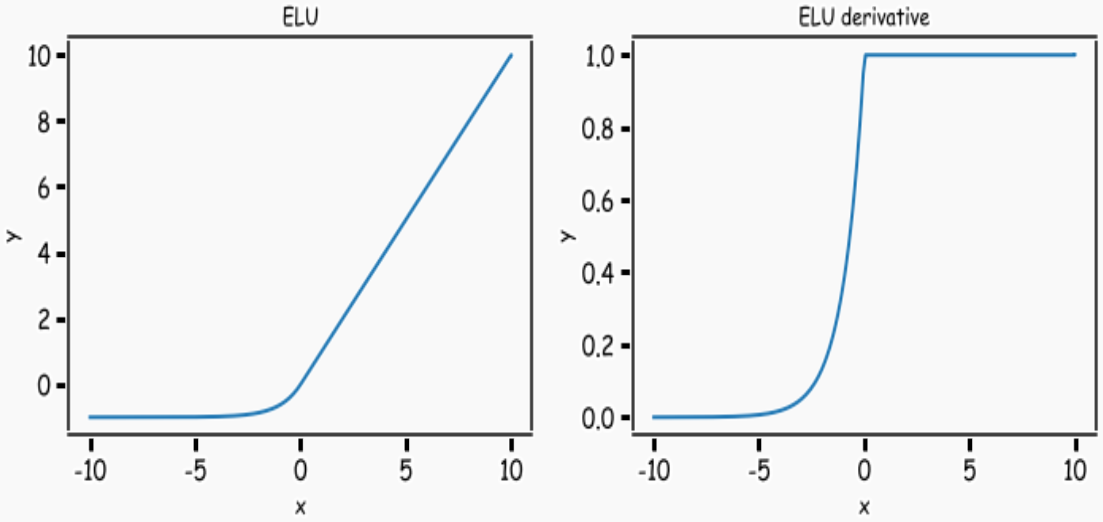
Derivative is **zero** for much of the domain. This leads to "vanishing gradients" in backpropagation.

Rectified Linear Unit, `ReLU()`, Exponential ReLU (ELU)

$$y = \max(0, x)$$



$$y = \max(0, x) + \alpha \min(0, e^x - 1)$$
where α takes a small value



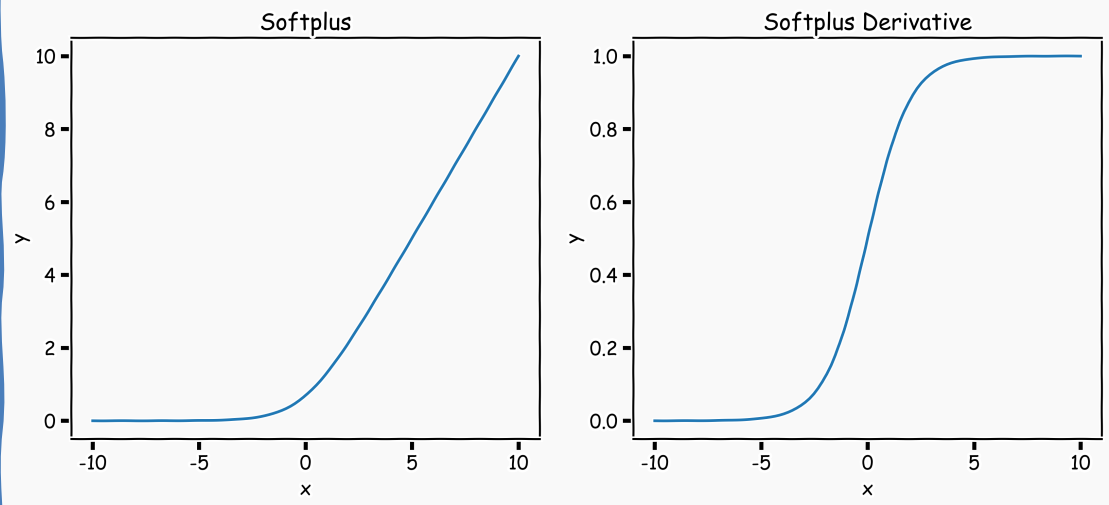
Two major advantages:

1. No vanishing gradient when x > 0
2. Provides sparsity (regularization) since y = 0 when x < 0

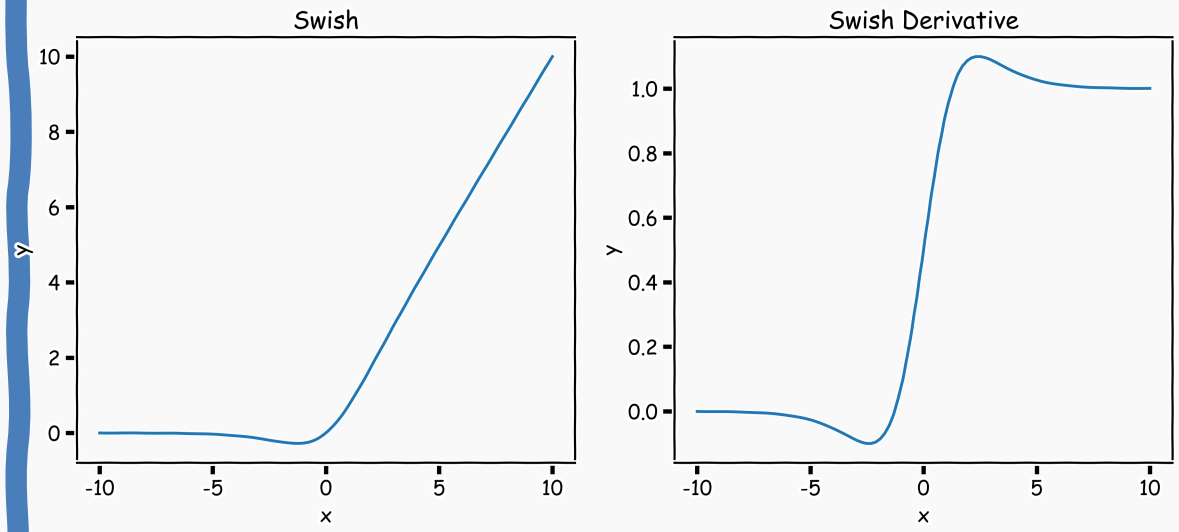
No vanishing gradients and easy to calculate.

Softplus and Swish

$$y = \log(1 + e^x)$$



$$g(x) = x\,\sigma(x)$$



The derivative of the softplus is the sigmoid logistic function, which is a smooth approximation of the derivative of the rectifier. So the derivative of the softplus is continuous.

Swish tends to work better than ReLU on deeper models across a number of challenging datasets.

TL;DR (too long; didn't read)

Use ReLU or leaky ReLU or ELU to start.

If you are concerned for a few points improvement in performance, experiment with swish and softplus and treat them as hyperparameters. In other words choose the activation that gives you the best validation loss.

Outline

Anatomy of a NN

Design choices

- Activation function
- **Loss function**
- Output units
- Architecture

Probabilistic modeling

Likelihood for a given measurement:

$$p(y_i|W;x_i)$$

Assume **independency**, likelihood for all measurements:

$$L(W;X,Y) = p(Y|W;X) = \prod_i p(y_i|W;x_i)$$

Maximize the likelihood, or equivalently minimizing the –ve log-likelihood:

$$\mathcal{L}(W;X,Y) = -\log L(W;X,Y) = \sum_i \log p(y_i|W;x_i)$$

Loss Function

Do not need to design separate loss functions if we follow the probabilistic modeling approach, i.e. minimize the –ve likelihood function.

Examples:

- Distribution is **Normal** then –ve log-likelihood is **MSE** :

$$p(y_i|W;x_i) = \frac{1}{\sqrt{\{2\pi^2\sigma\}}} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}$$

$$\mathcal{L}(W;X,Y) = \sum_i (y_i - \hat{y}_i)^2$$

- Distribution is **Bernouli** then –ve log-likelihood is **Binary Cross-Entropy**:

$$p(y_i|W;x_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

$$\mathcal{L}(W;X,Y) = -\sum_i [y_i \log p_i + (1-y_i)\log(1-p_i)]$$

Loss Function

- For y that follow **Multinouli,** then likelihood is:

$$p(y_i|W;x_i) = \prod_k p\,(y_i = k)^{\mathbb{I}(y_i=k)}$$

Cross-Entropy

$$\mathcal{L}(W;X,Y) = -\sum_i \sum_k \mathbb{I}(y_i = k) \log p(y_i = k)$$

where k is the class and $\mathbb{I}(y_i = k)$ is the indicator function.

Questions:

- How do we know what distribution to use?
- Why not MSE for classification?

Design Choices

Activation function

Loss function

Output units

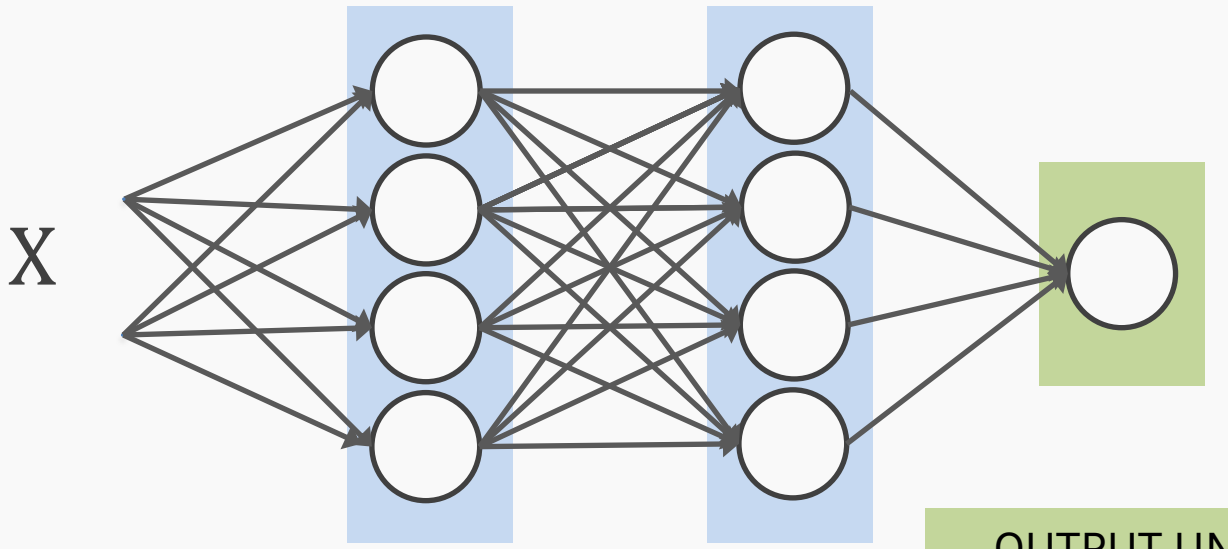
Architecture

Optimizer

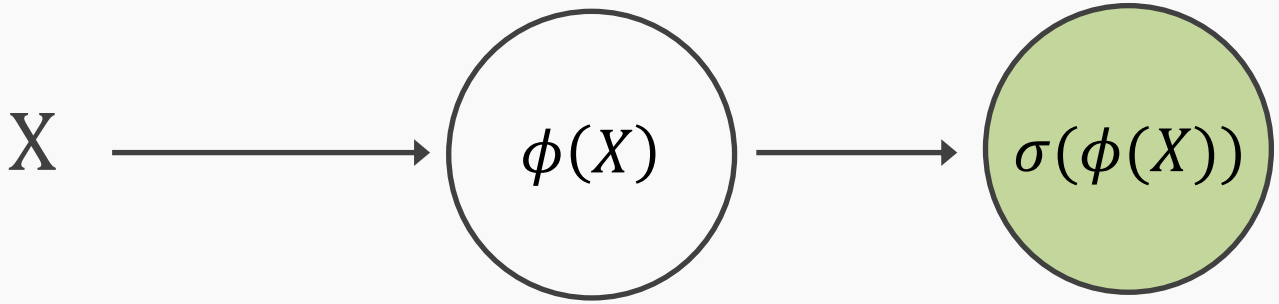
Output Units

Output Type	Output Distribution	Output layer	Loss Function
Binary	Bernoulli	?	Binary Cross Entropy

Output unit for binary classification



$$\hat{Y} = \mathrm{P}(\mathrm{y} = 1)$$



$$\hat{Y} = \mathrm{P}(\mathrm{y} = 1)$$

$$X \Longrightarrow \phi(X) \Longrightarrow P(y = 1) = \frac{1}{1 + e^{-\phi(X)}}$$

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete			

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinouli		

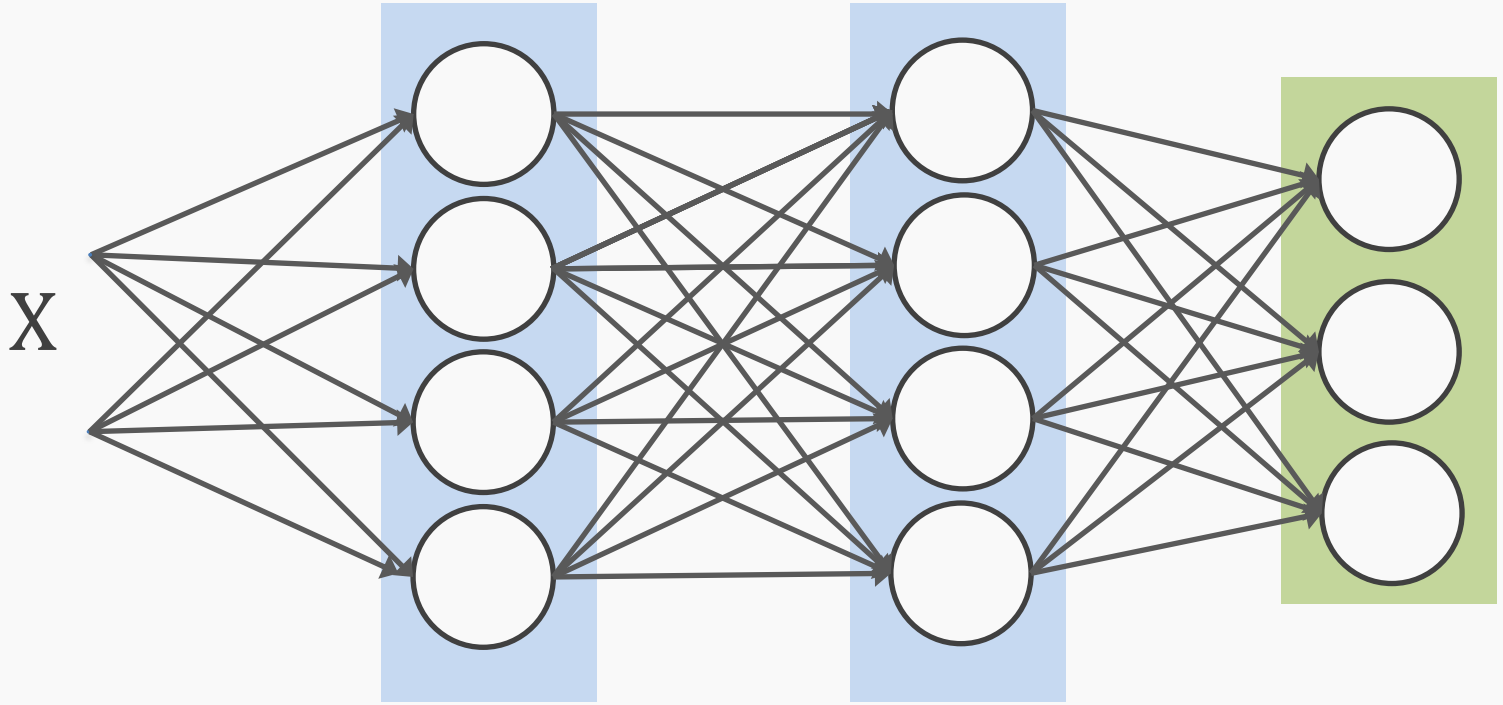
Output Units

Output Type	Output Distribution	Output layer	Loss Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinouli		Cross Entropy

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinouli	?	Cross Entropy

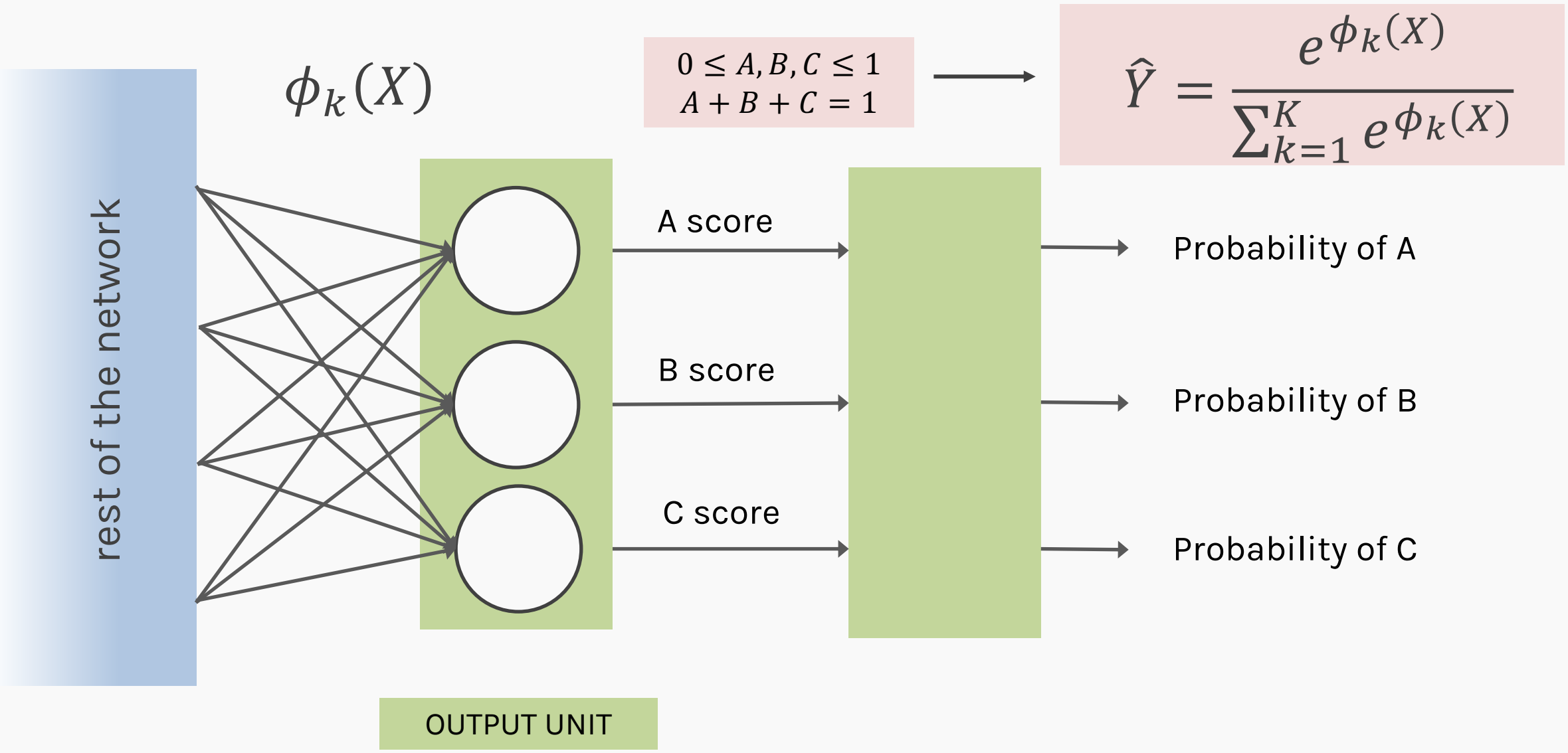
Output unit for multi-class classification



$$\hat{Y} = [P_1, P_2, P_3]$$

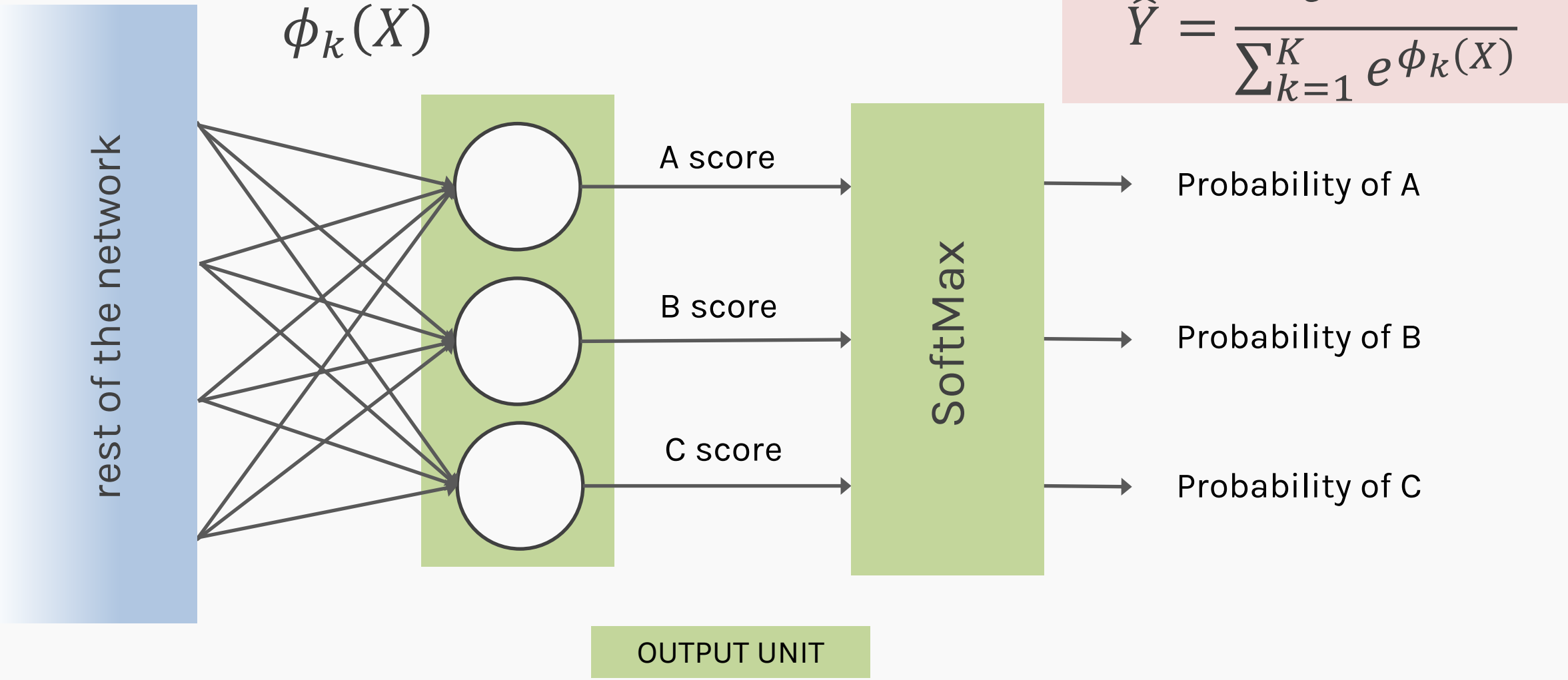
OUTPUT UNIT

SoftMax


$\phi_k(X)$
$0 \le A, B, C \le 1$
$A + B + C = 1$
$\hat{Y} = \frac{e^{\phi_k(X)}}{\sum_{k=1}^{K} e^{\phi_k(X)}}$
rest of the network
A score
B score
C score
Probability of A
Probability of B
Probability of C
OUTPUT UNIT

SoftMax

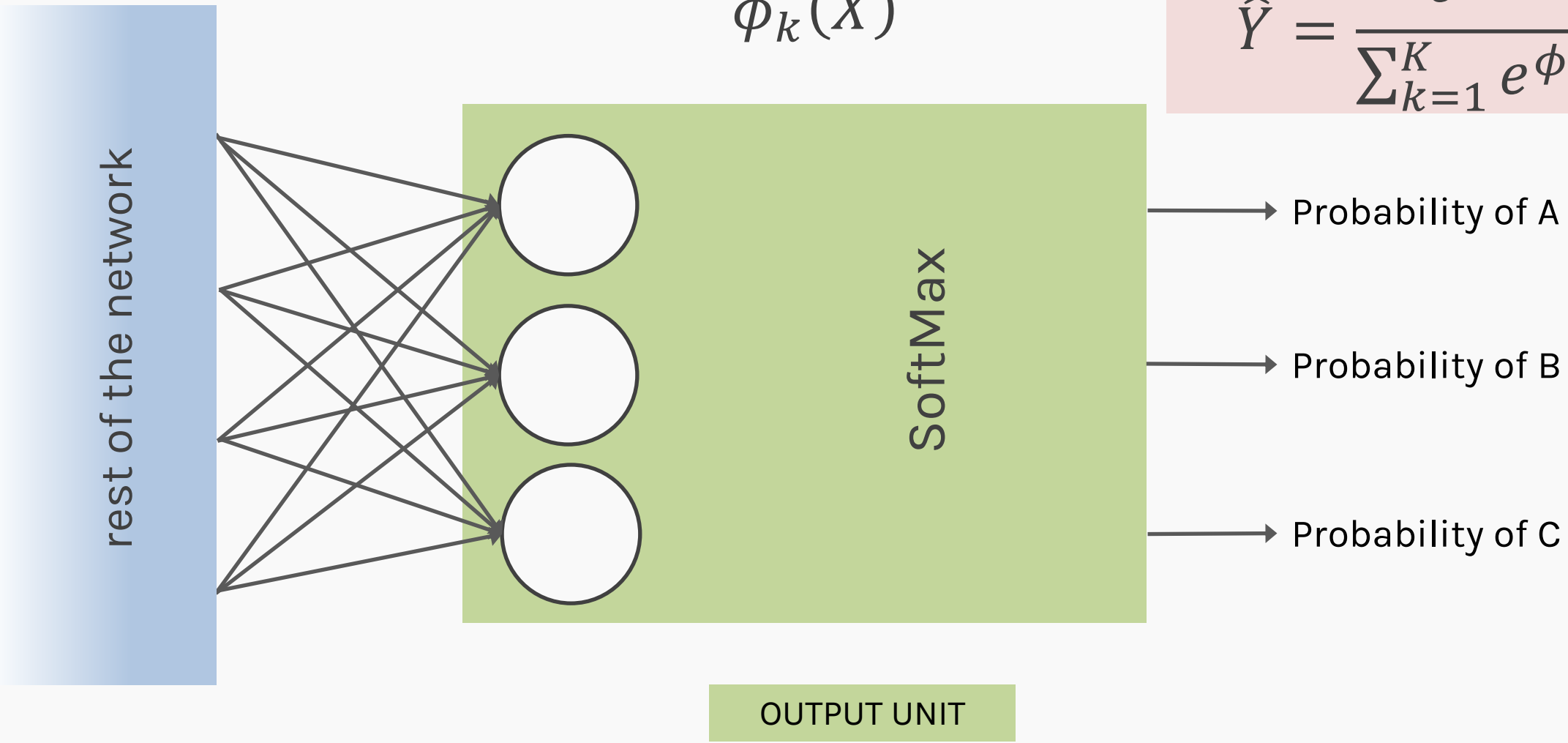
$$\hat{Y} = \frac{e^{\phi_k(X)}}{\sum_{k=1}^{K} e^{\phi_k(X)}}$$



SoftMax

$$\phi_k(X)$$

$$\hat{Y} = \frac{e^{\phi_k(X)}}{\sum_{k=1}^{K} e^{\phi_k(X)}}$$



Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous			

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian		

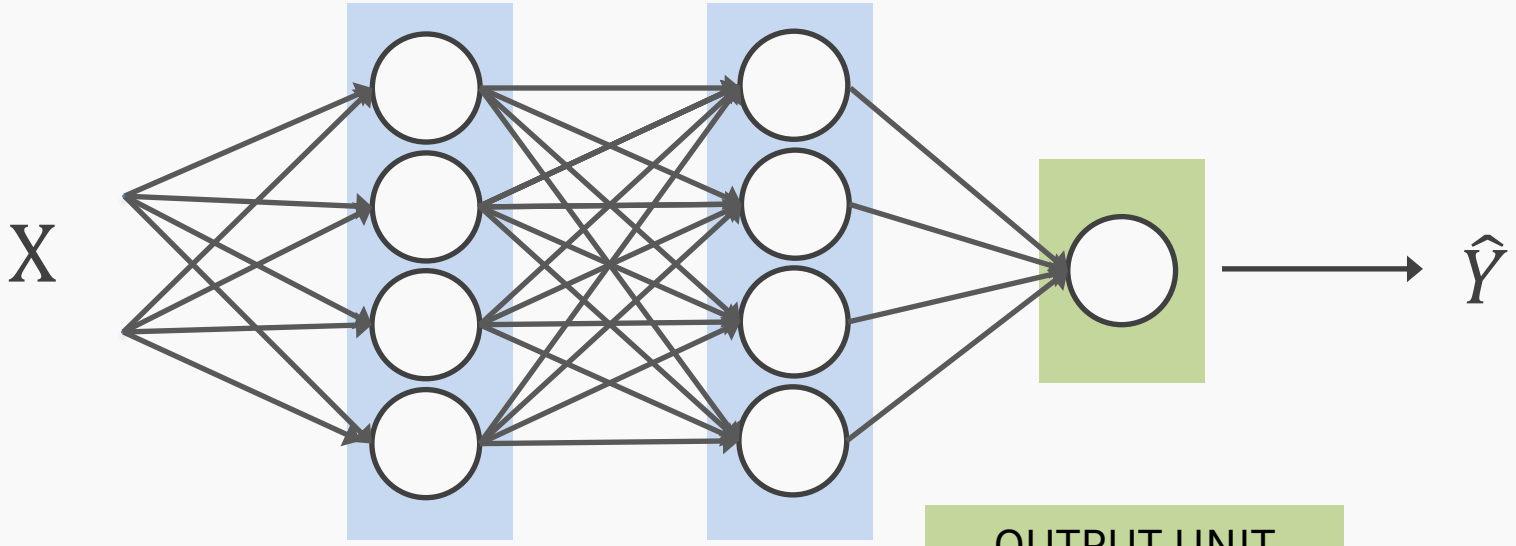
Output Units

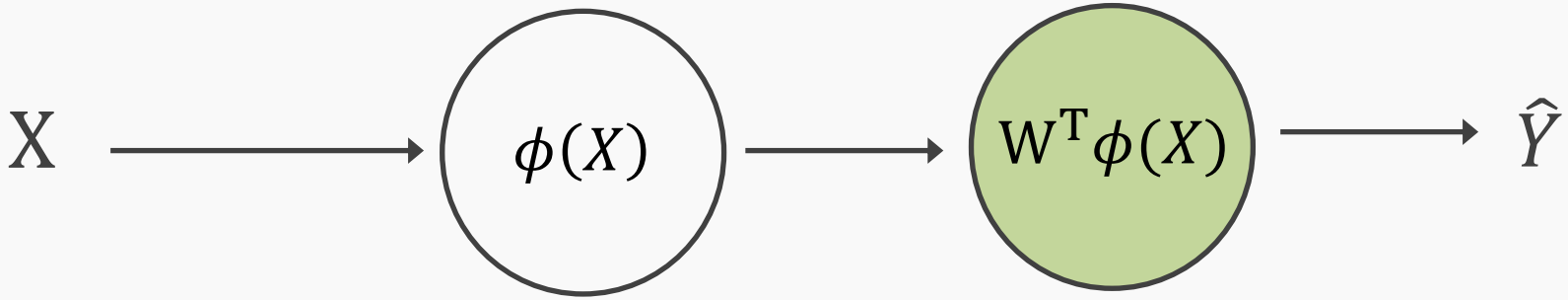
Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian		MSE

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	?	MSE

Output unit for regression





$$X \Longrightarrow \phi(X) \Longrightarrow \hat{Y} = W^T\phi(X)$$

Output Units

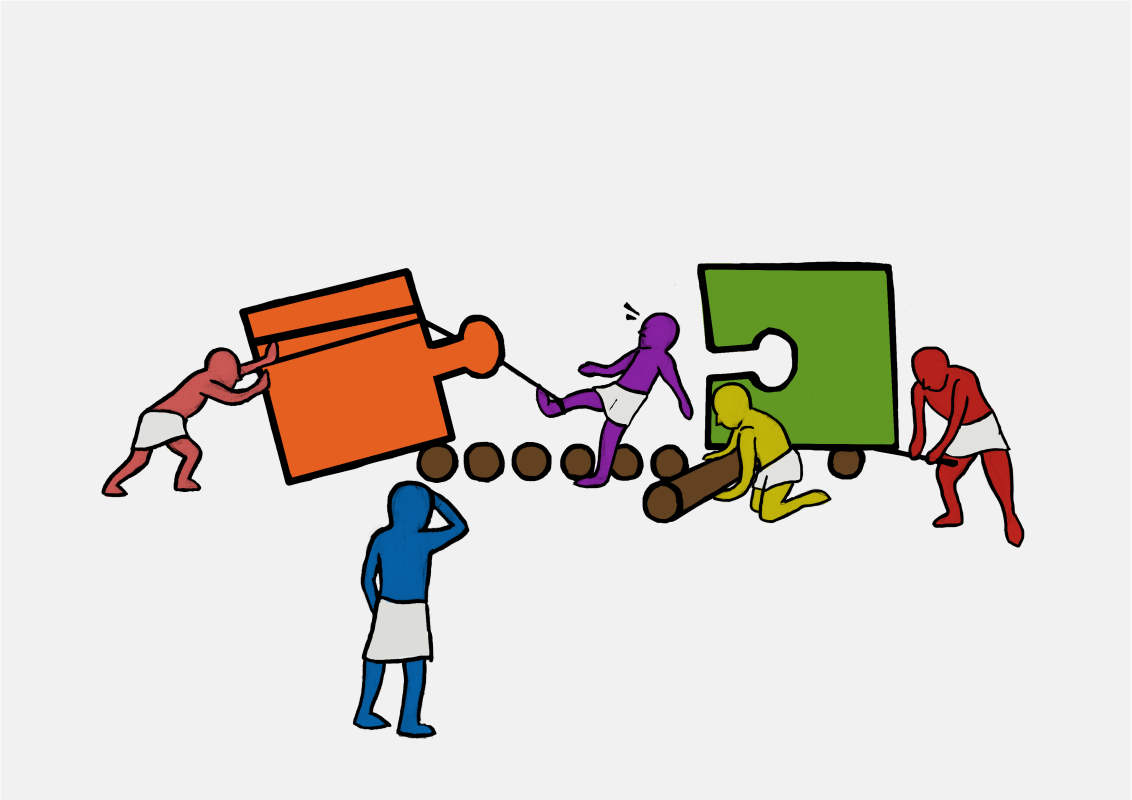
Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	Linear	MSE

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	Linear	MSE
Continuous	Arbitrary	-	

Output Units

Output Type	Output Distribution	Output layer	Cost Function
Binary	Bernoulli	Sigmoid	Binary Cross Entropy
Discrete	Multinoulli	Softmax	Cross Entropy
Continuous	Gaussian	Linear	MSE
Continuous	Arbitrary	-	GANS



Constructing an MLP with Keras