

Today

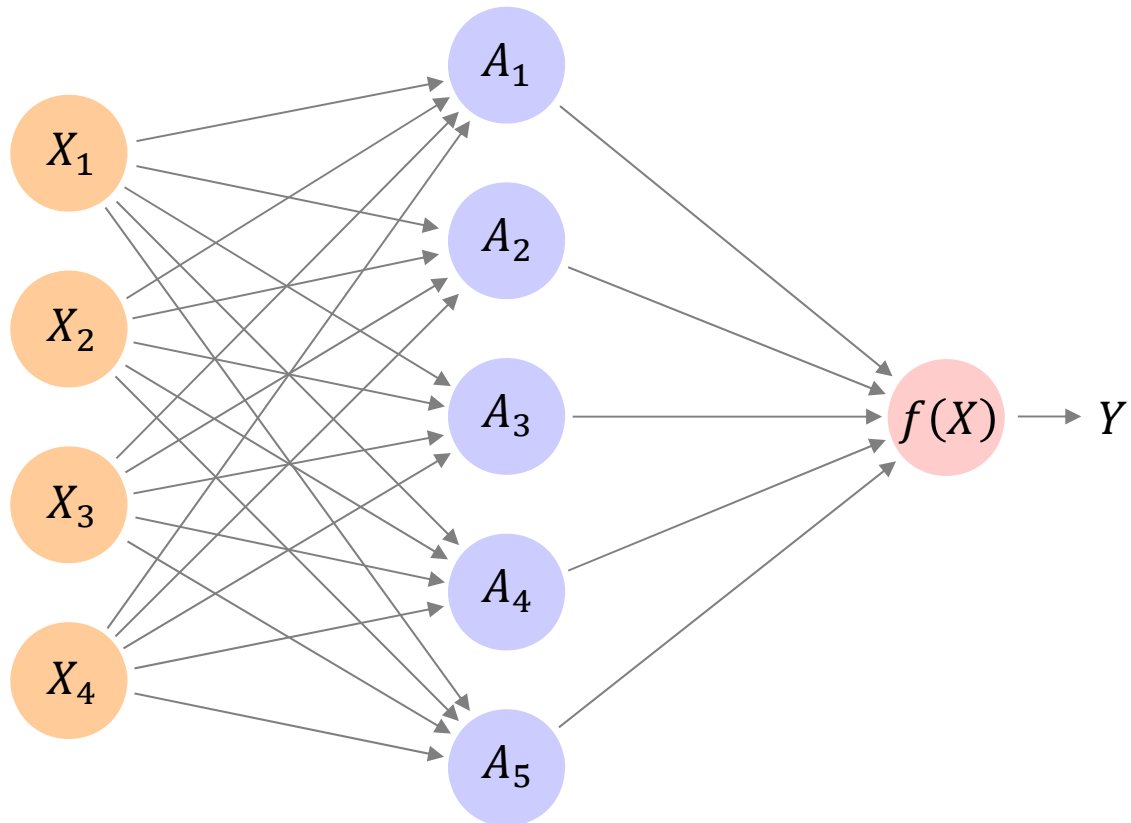
- Neural networks and deep learning
 - Single layer neural networks
 - Multi-layer neural networks
 - Convolutional neural networks
 - U-net
 - Transformers

Single layer NN

Input
layer

Hidden
layer

Output
layer

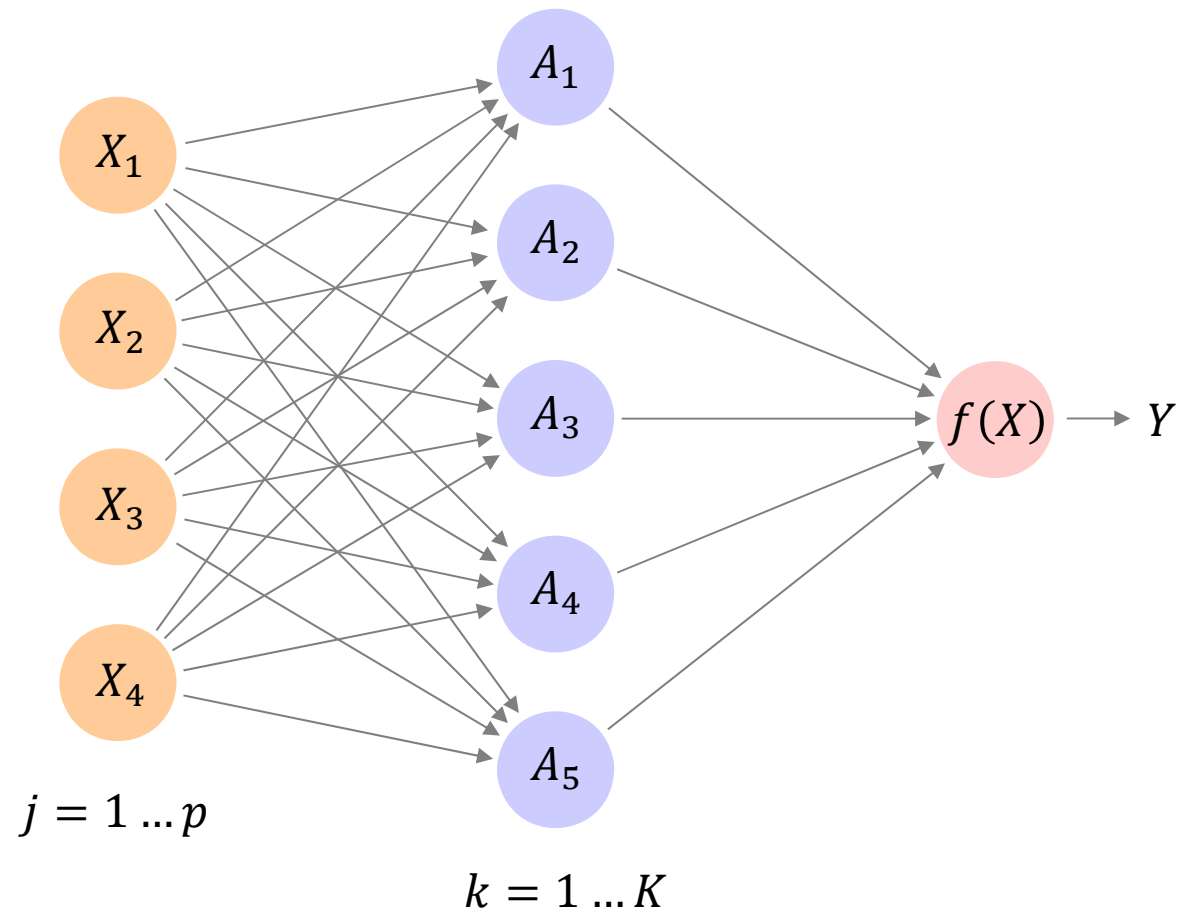


Single layer NN

Input
layer

Hidden
layer

Output
layer

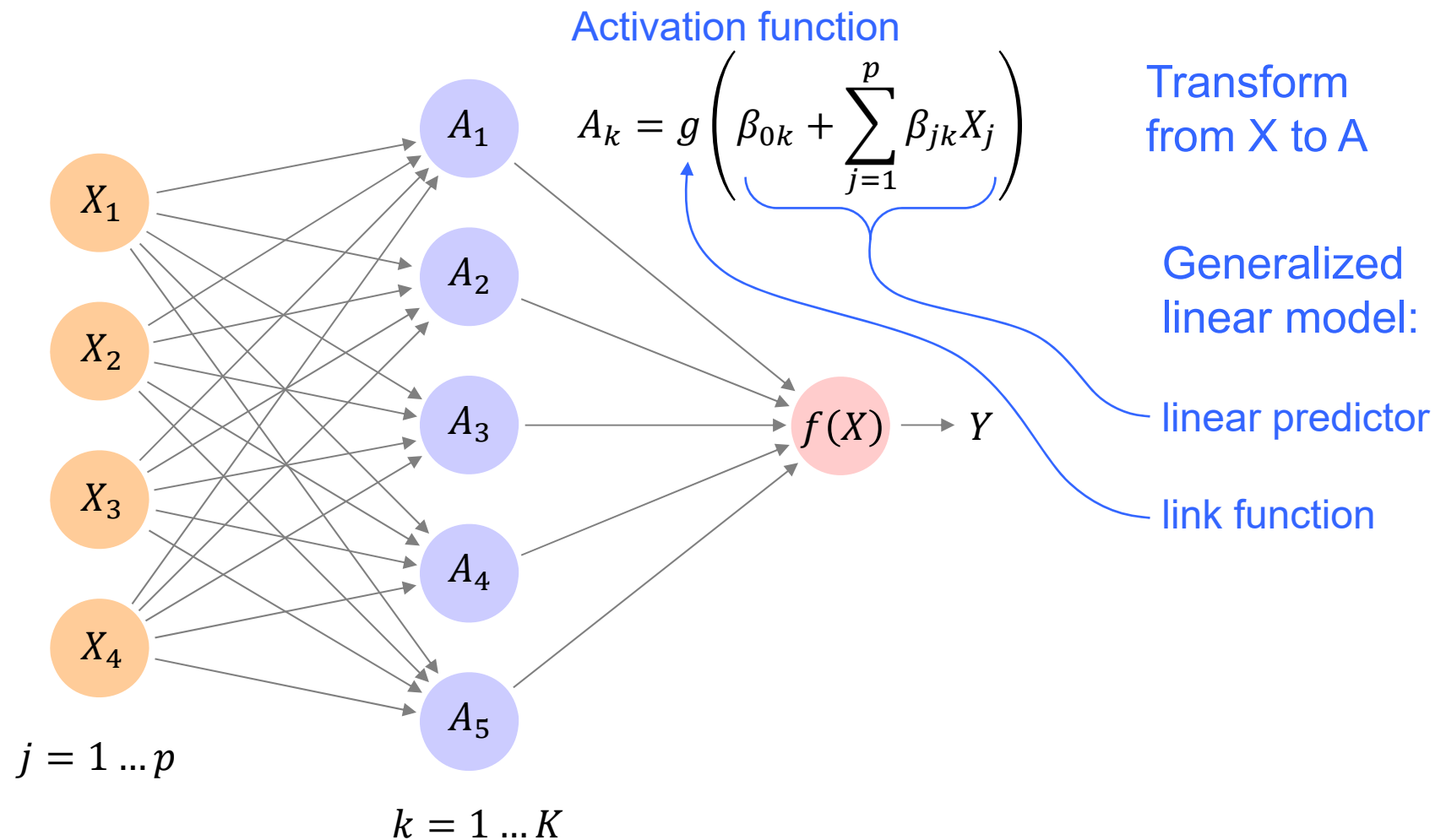


Single layer NN

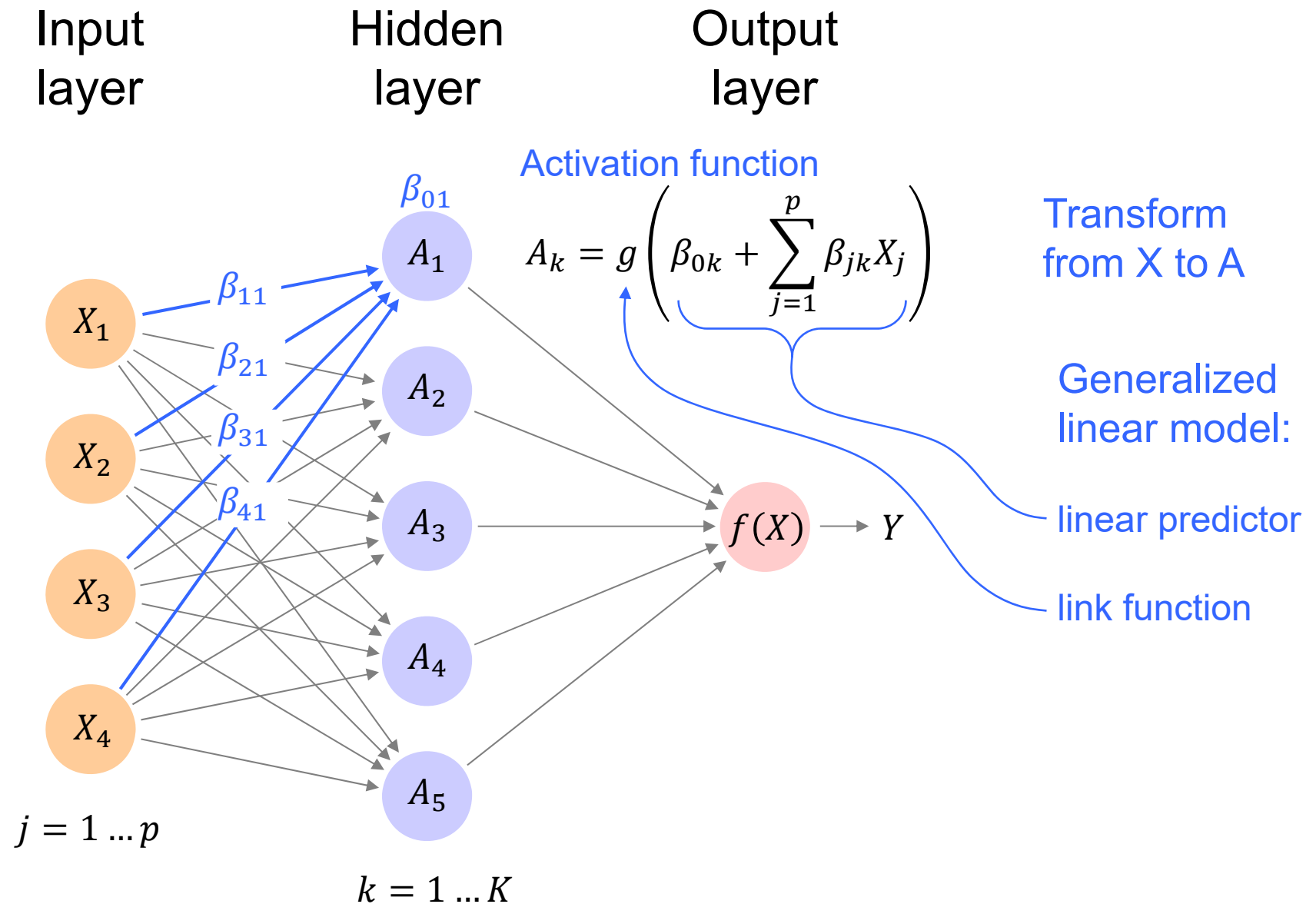
Input
layer

Hidden
layer

Output
layer



Single layer NN

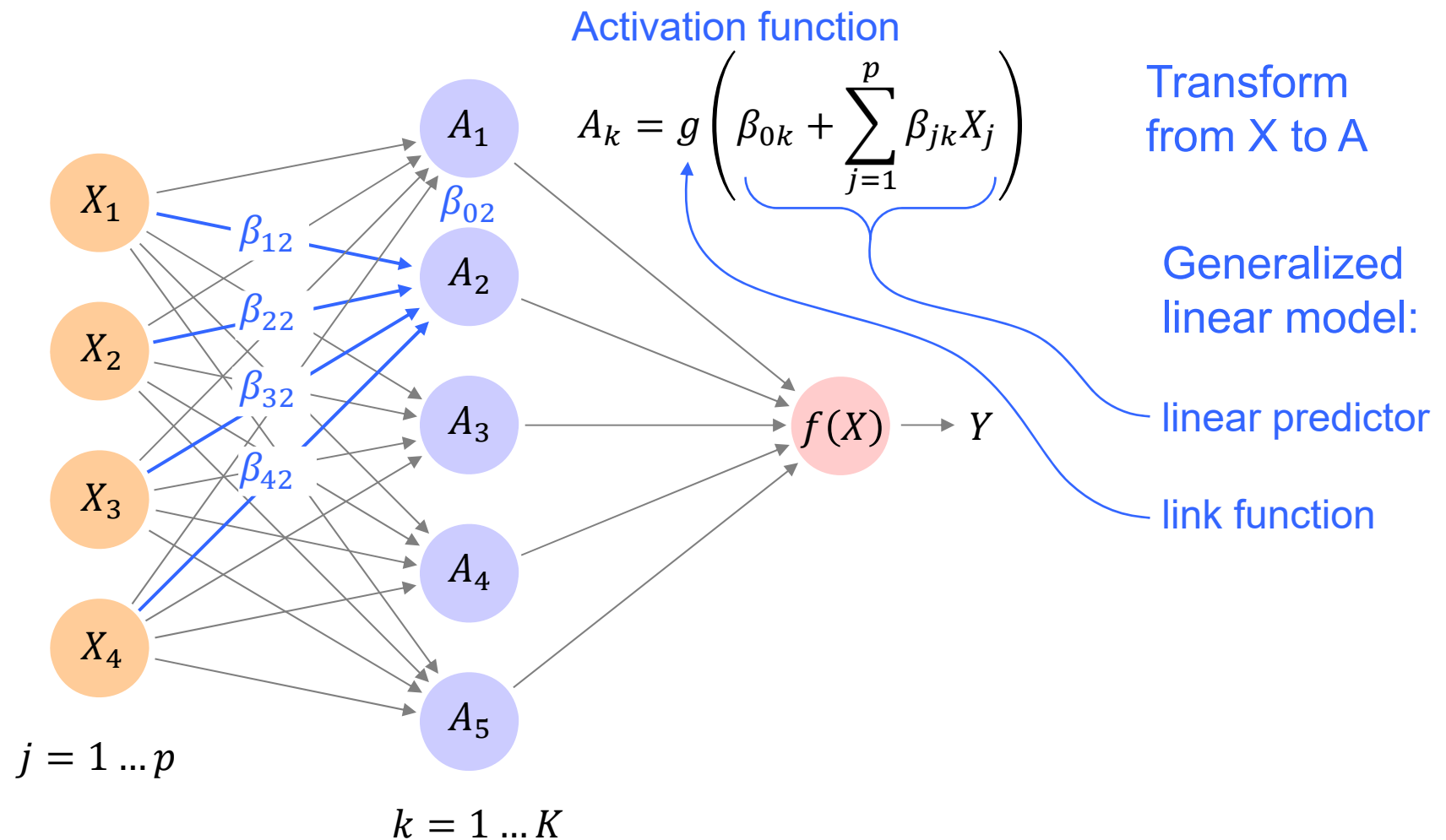


Single layer NN

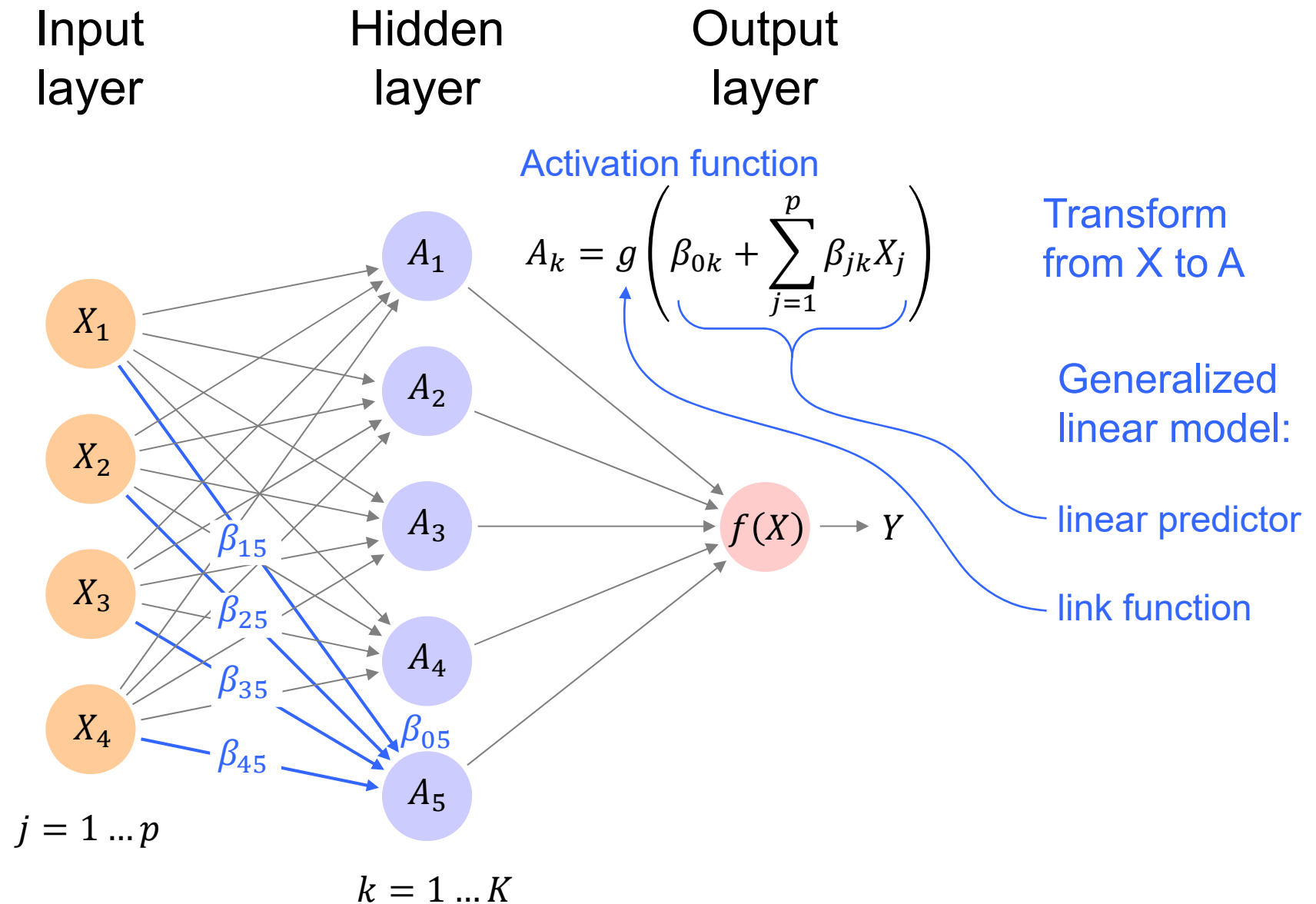
Input
layer

Hidden
layer

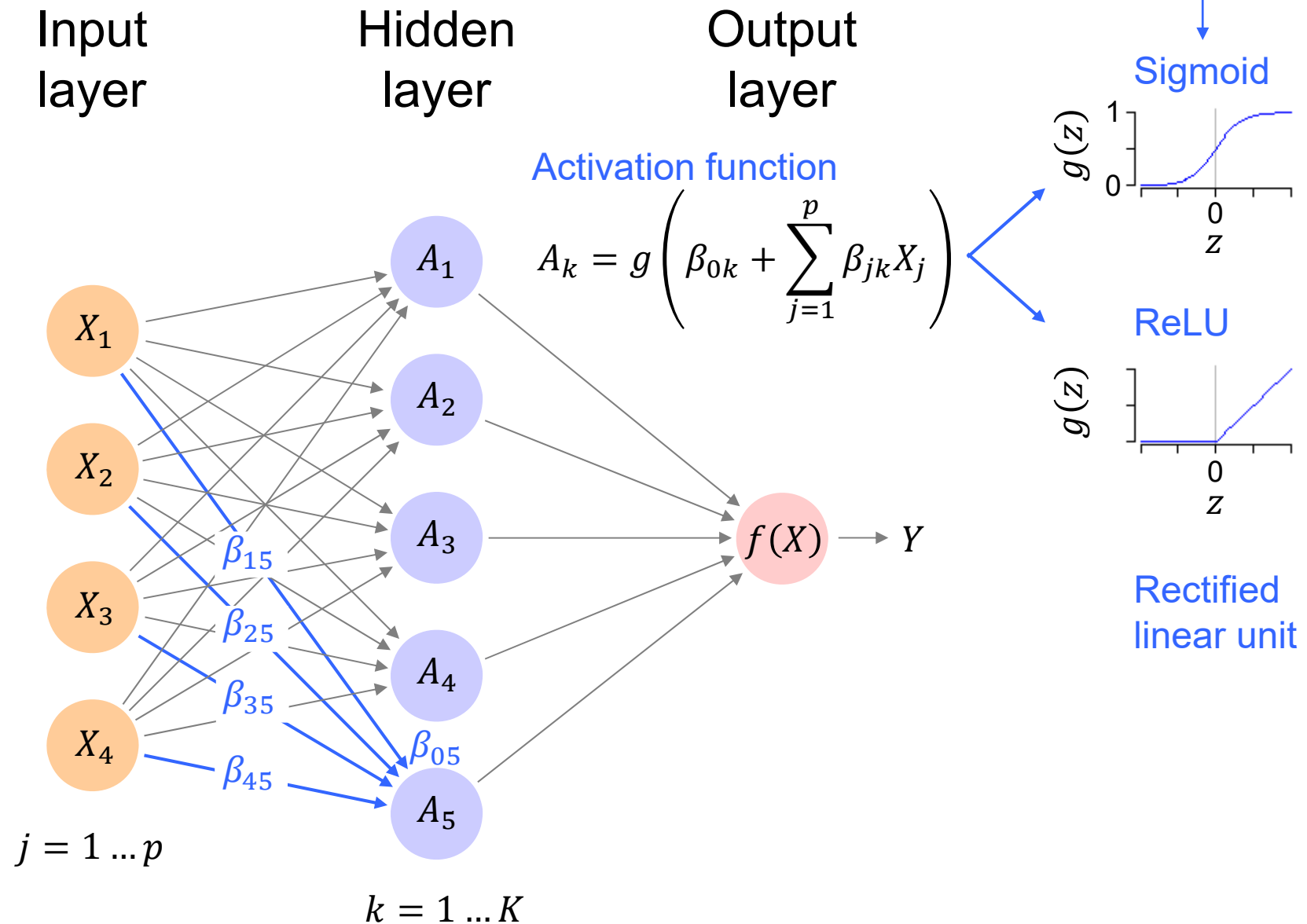
Output
layer



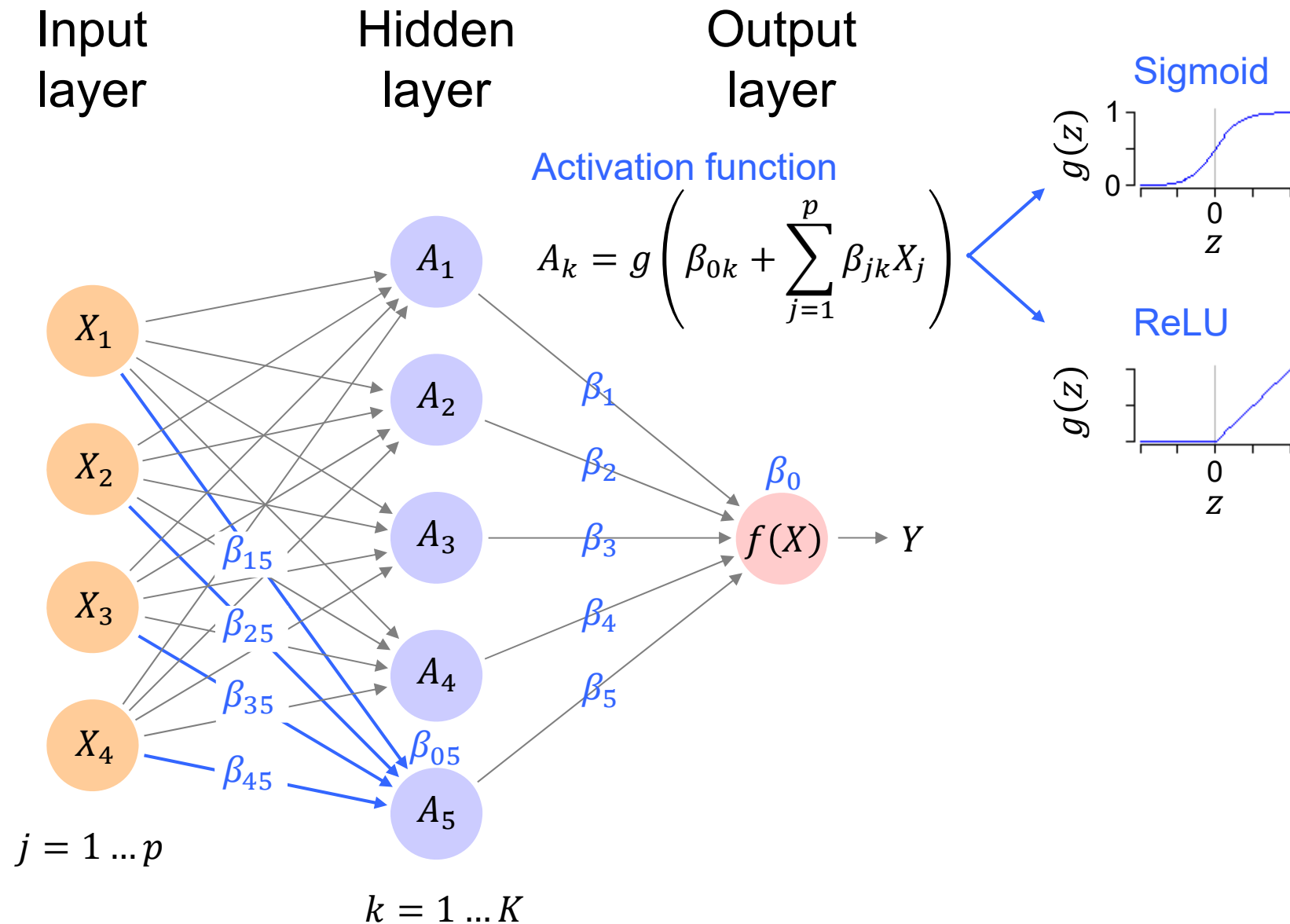
Single layer NN



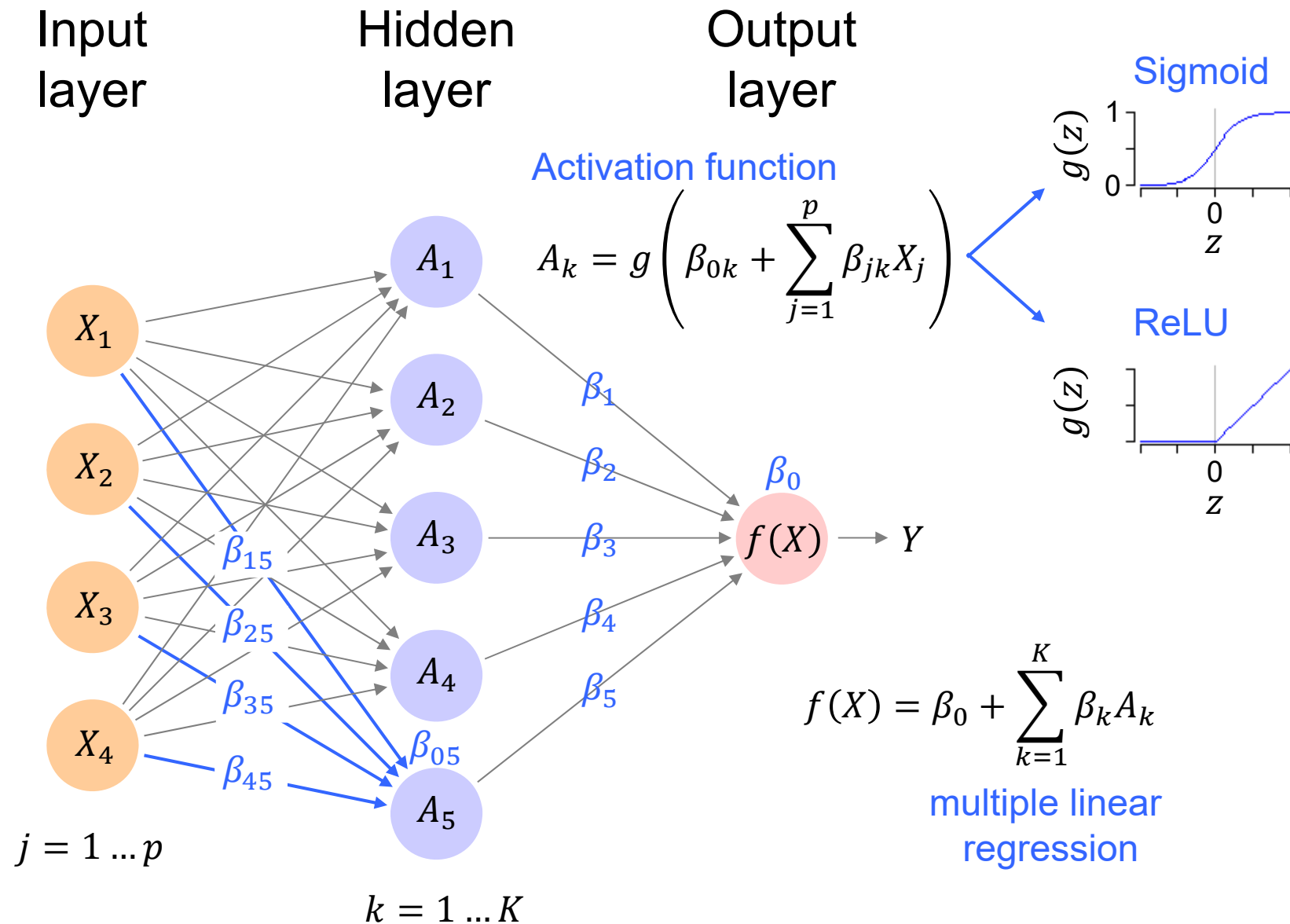
Single layer NN



Single layer NN



Single layer NN



Single layer NN

Model algorithm

define $g(z)$

load x_j

set K

set parameters: $\beta_{..}$

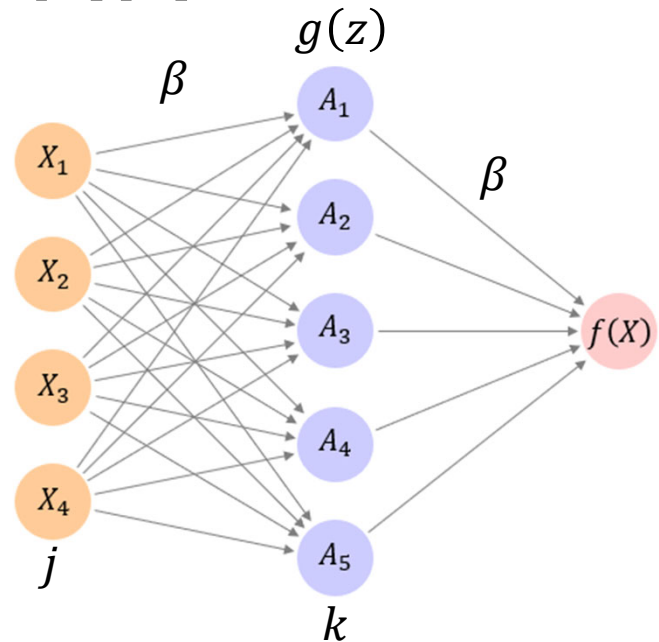
for each activation unit k in $1:K$

 calculate linear predictor: $z_k = \beta_{0k} + \sum_j \beta_{jk} x_j$

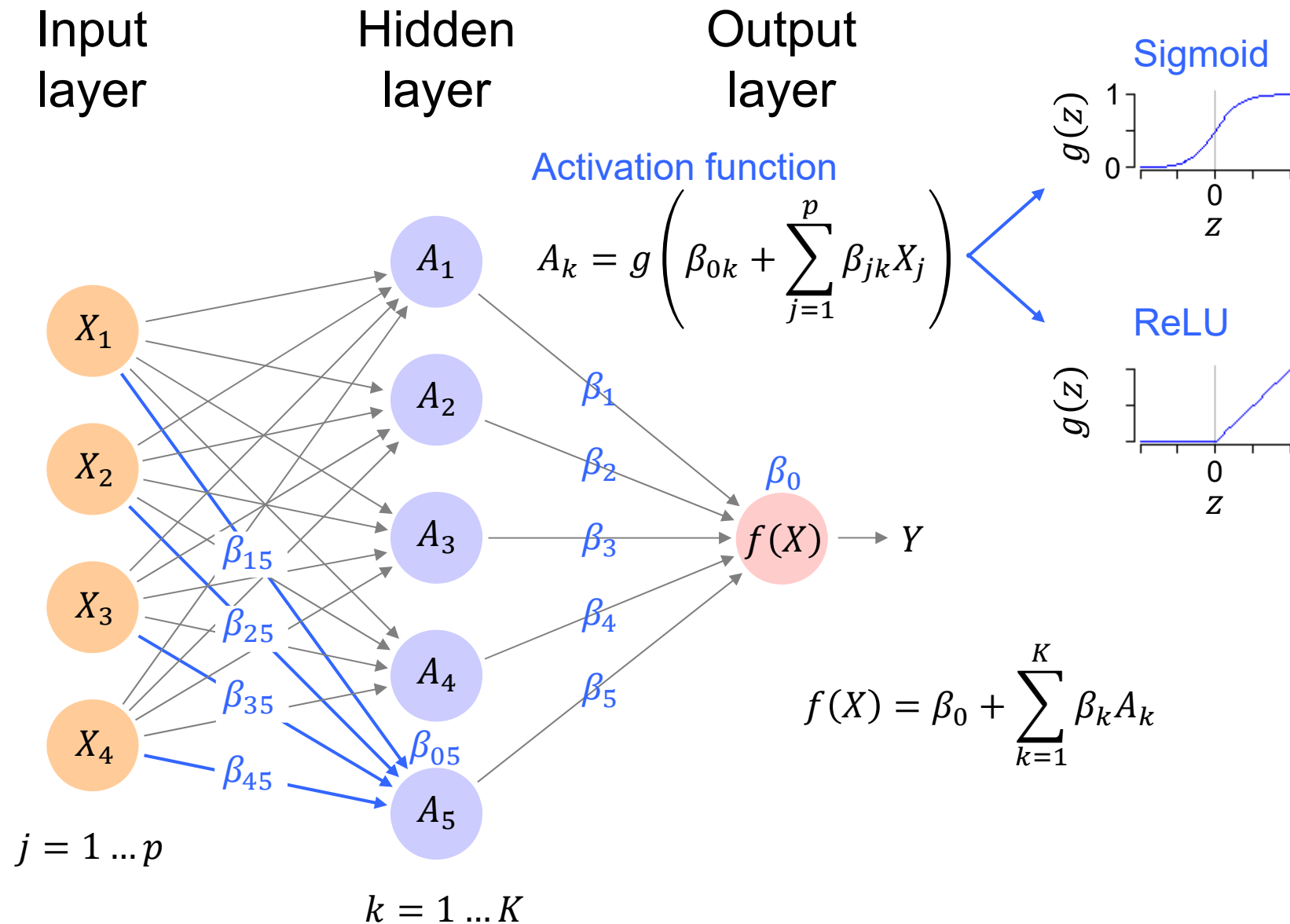
 calculate nonlinear activation: $A_k = g(z_k)$

calculate linear model: $f(x) = \beta_0 + \sum_k \beta_k A_k$

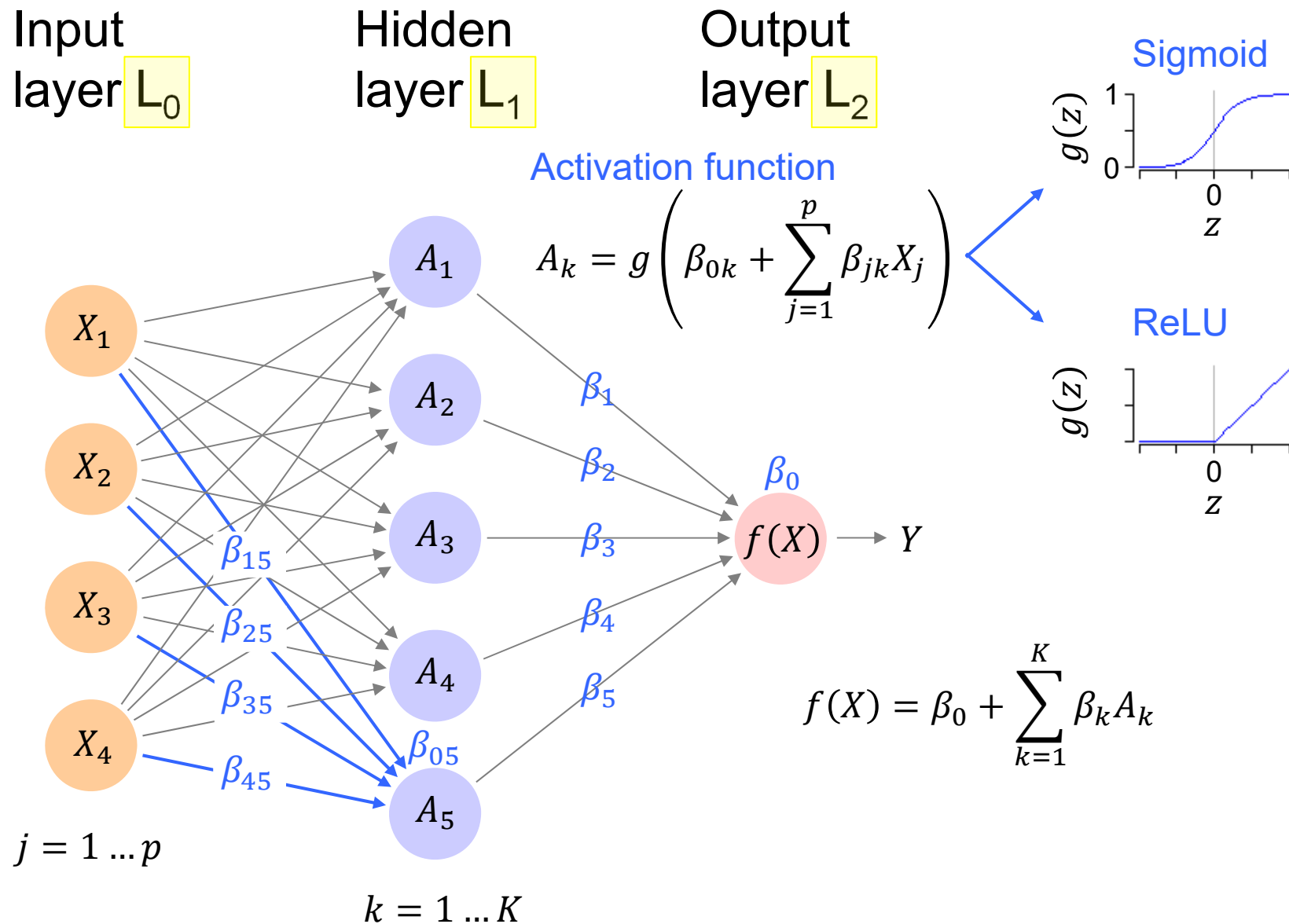
return $f(x)$



Single layer NN (generalizing)



Single layer NN (generalizing)

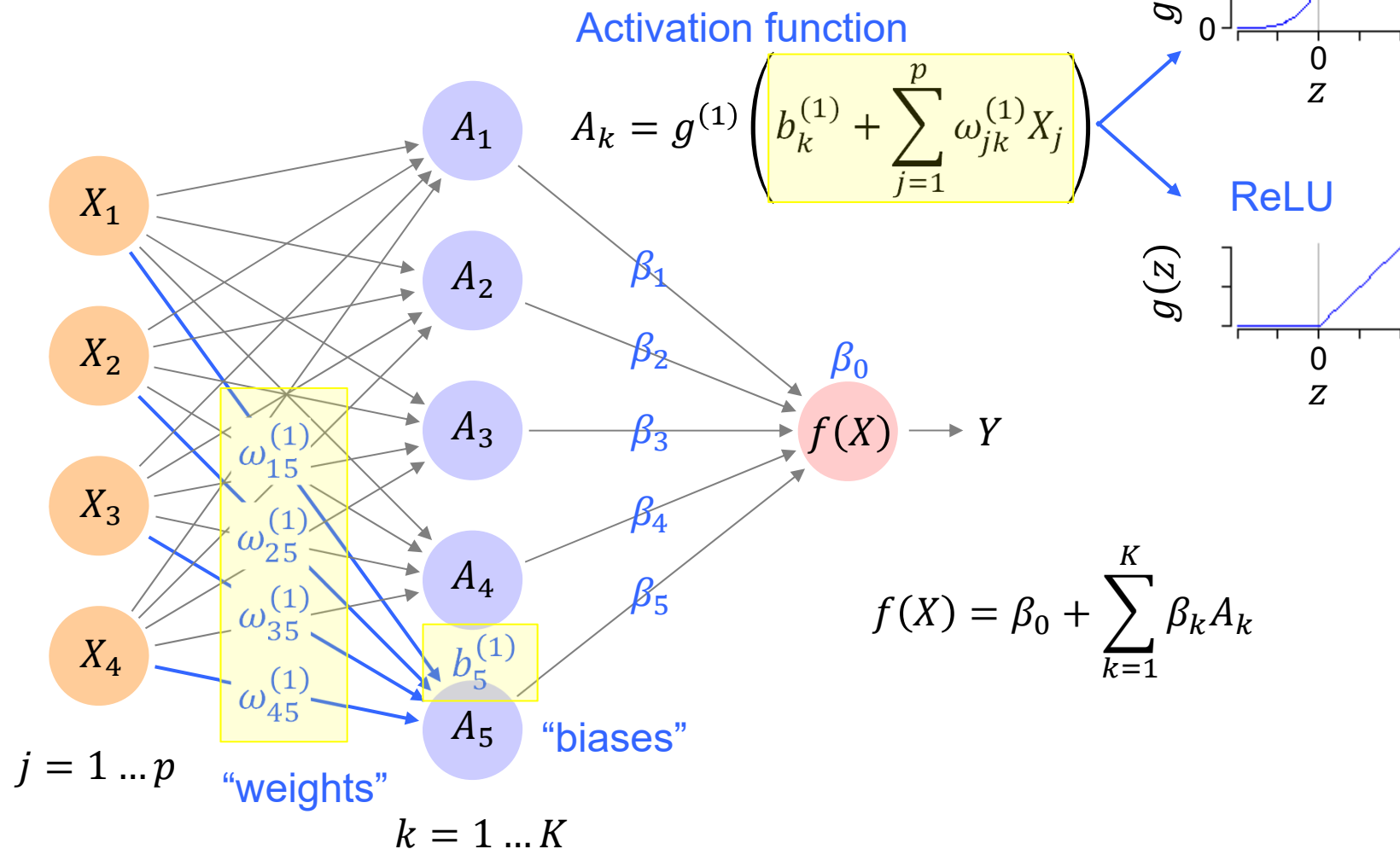


Single layer NN (generalizing)

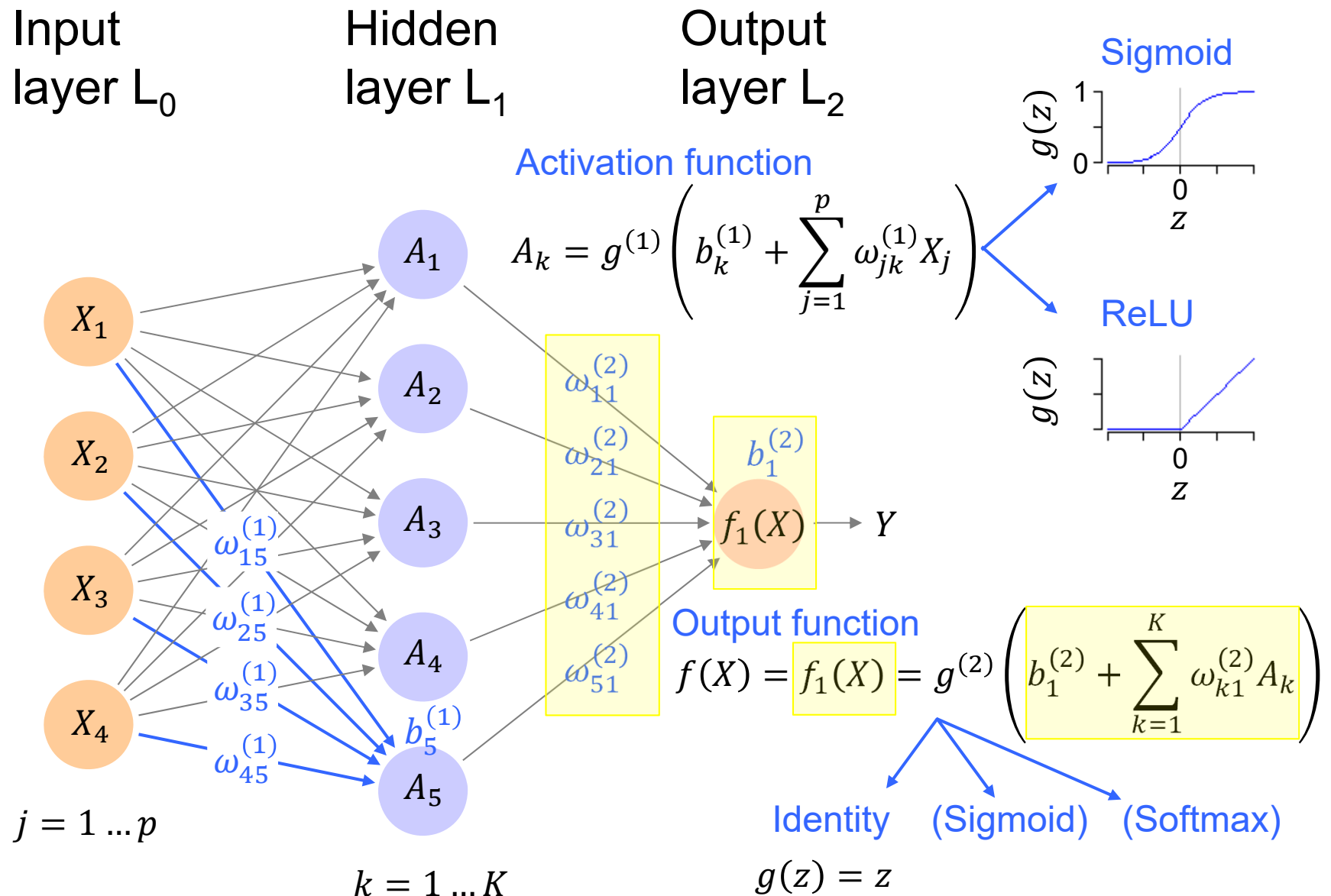
Input
layer L_0

Hidden
layer L_1

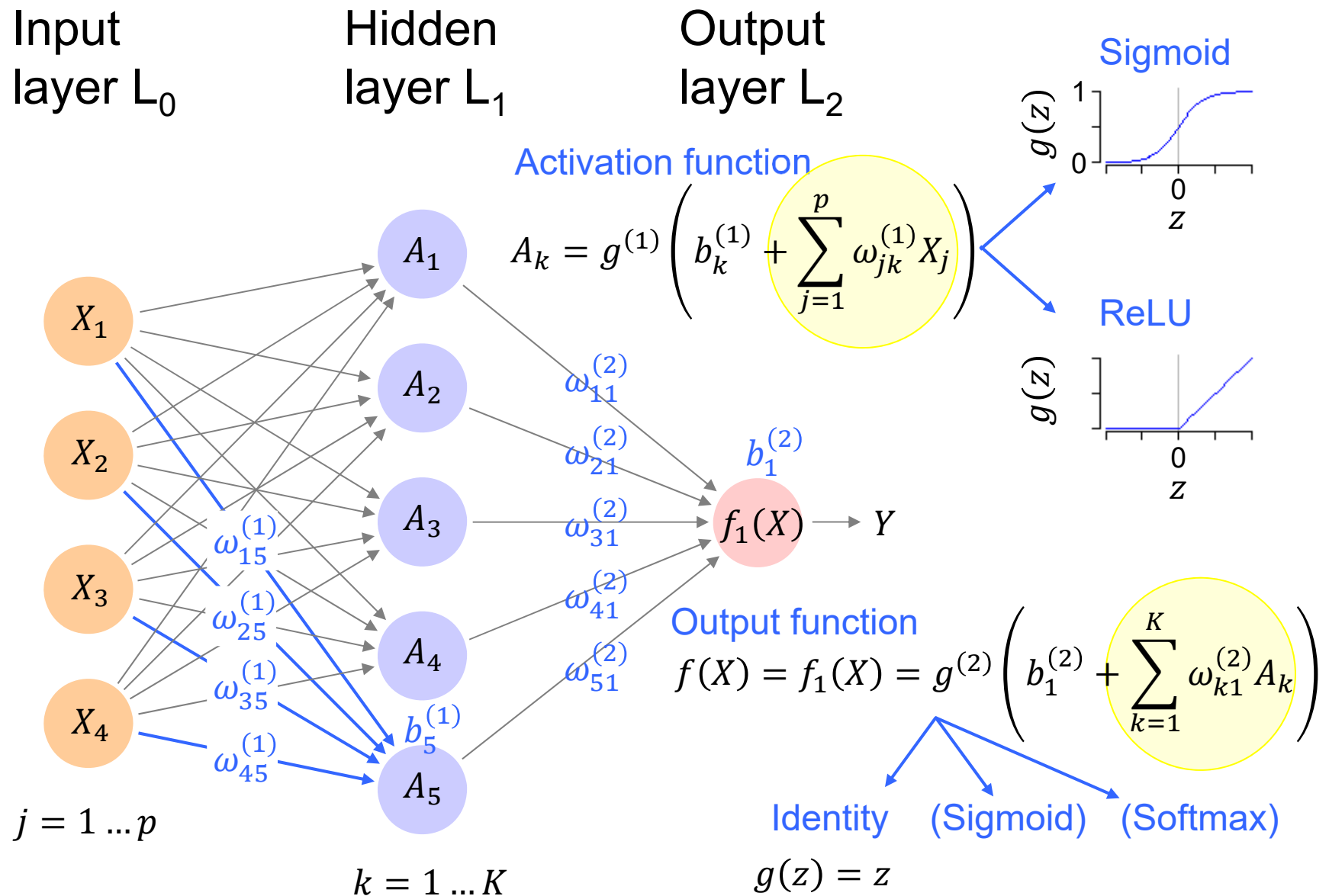
Output
layer L_2



Single layer NN (generalizing)



Single layer NN (generalizing)



Single layer NN (generalizing)

$$\sum_{j=1}^p \omega_{jk}^{(1)} X_j$$

data rows
 $i = 1 \dots n$

multiply down columns
then add across rows

$$\begin{array}{cccc} \downarrow & \downarrow & \downarrow & \downarrow \\ \omega_{1k} & \omega_{2k} & \omega_{3k} & \omega_{4k} \\ * & + & * & + & * & + & * \\ X_1 & X_2 & X_3 & X_4 \\ \\ x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & x_{n4} \end{array}$$

data columns
 $j = 1 \dots p$

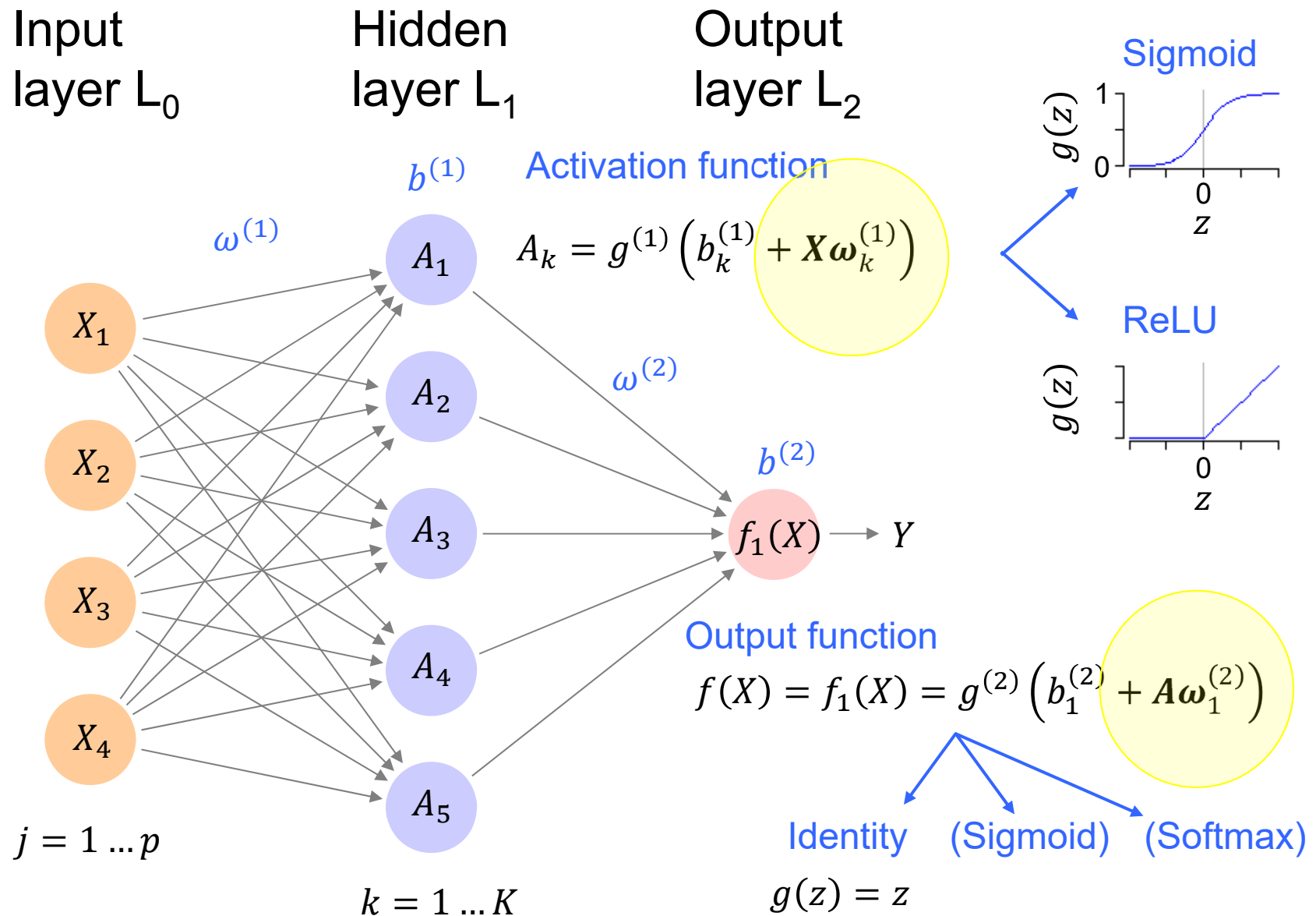
Matrix multiplication

$$\mathbf{X} \boldsymbol{\omega}_k$$

$$n \times p \quad p \times 1$$

$$\mathbf{R}: \quad \times \quad \% \quad * \quad \% \quad \mathbf{W}$$

Single layer NN (generalizing)



Single layer NN

Model algorithm

define $g(z)$

load and prepare x_j

set K

set $\omega_{jk}^{(1)}$, $b_k^{(1)}$, $\omega_{k1}^{(2)}$, $b_1^{(2)}$

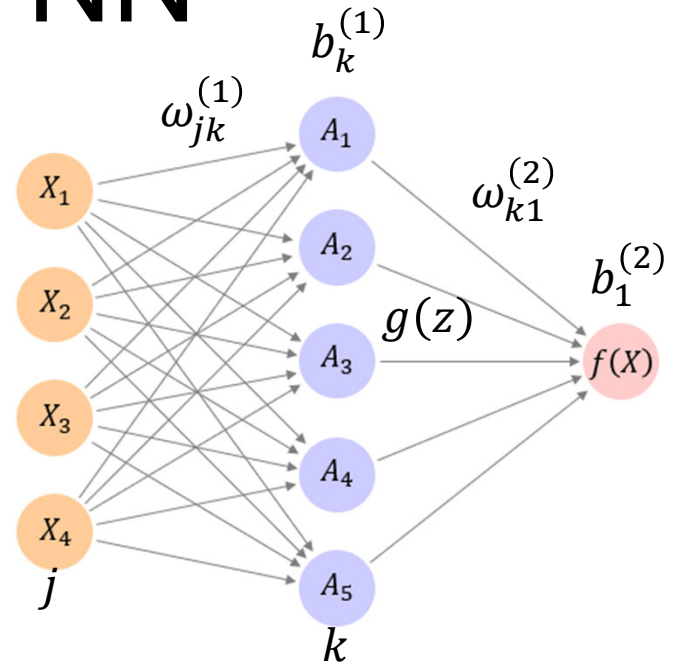
for each activation unit k in $1:K$

calculate linear predictor: $z_k = b_k^{(1)} + \mathbf{X}\boldsymbol{\omega}_k^{(1)}$

calculate nonlinear activation: $A_k = g(z_k)$

calculate linear model: $f(x) = b_1^{(2)} + \mathbf{A}\boldsymbol{\omega}_1^{(2)}$

return $f(x)$



Training algorithm

Loss function (MSE)

$$\text{MSE}(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta}(x_i))^2$$

$$\theta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{10} \\ \beta_{11} \\ \vdots \end{pmatrix}$$

Training algorithm

Stochastic gradient descent

guess θ (typically random)

set λ (learning rate)

for iterations (e.g. until $\text{MSE}(\theta)$ stops decreasing)

 randomly sample the data

 calculate gradient of $\text{MSE}(\theta)$: $\frac{\delta \text{MSE}(\theta)}{\delta \theta}$ Method: back propagation

$$\theta \leftarrow \theta - \lambda \frac{\delta \text{MSE}(\theta)}{\delta \theta}$$

Training algorithm

Stochastic gradient descent (mini batch)

guess θ (typically random)

set λ (learning rate)

for many epochs

 randomly partition data into batches

 for each batch

 calculate gradient of $\text{MSE}(\theta)$: $\frac{\delta \text{MSE}(\theta)}{\delta \theta}$

$$\theta \leftarrow \theta - \lambda \frac{\delta \text{MSE}(\theta)}{\delta \theta}$$