# 4. Deep learning

# Outline

# From linear to non-linear models

- In linear regression, the prediction

$$h(x;\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n,$$

  is a linear (strictly speaking, affine) function of the feature vector $x$.

- In logistic regression,

$$\log \frac{P(y=1|x;\theta)}{1 - P(y=1|x;\theta)} = h(x;\theta)$$

- Such linear models are simple and easy to interpret, but not flexible enough to capture complex patterns in various applications.

- It is desirable to make $h(x;\theta)$ a non-linear function of $x$.

# Non-linear models

- Many choices of non-linear functions for $h(x;\theta)$, e.g.:
  - Polynomial regression
  - Kernel methods
  - Decision trees, random forests, boosting
  - Neural networks
- Models based on neural networks have proven to be very effective.
- Besides neural network architectures, various computing techniques are developed to train them.
- The combinations of neural network architectures and computing techniques are often referred to as **deep learning**.

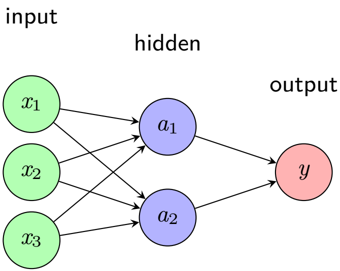
# Outline

Non-linear models

Neural networks

Layers in neural networks

# Neural networks




- values of hidden units

$$a_1 = \mathrm{ReLU}(w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + w_{13}^{[1]} x_3 + b_1^{[1]})$$
$$a_2 = \mathrm{ReLU}(w_{21}^{[1]} x_1 + w_{22}^{[1]} x_2 + w_{23}^{[1]} x_3 + b_2^{[1]})$$

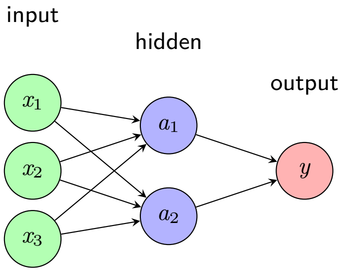
- value of output unit
  - regression:

$$y = w_1^{[2]} a_1 + w_2^{[2]} a_2 + b^{[2]}$$

  - classification:

$$y = \mathrm{Sigmoid}(w_1^{[2]} a_1 + w_2^{[2]} a_2 + b^{[2]})$$

# Neural networks in matrix notation




- values of hidden units

$$a = \mathrm{ReLU}(W^{[1]}x + b^{[1]})_1$$

- value of output unit
  - regression:

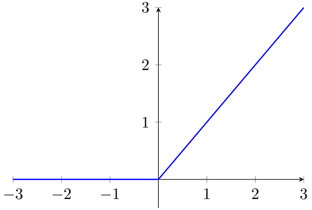
$$y = W^{[2]}a + b^{[2]}$$

  - classification:

$$y = \mathrm{Sigmoid}(W^{[2]}a + b^{[2]})$$

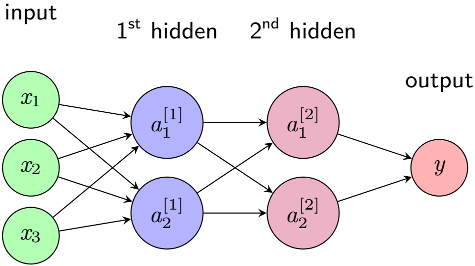
# Activation functions

- Used to introduce non-linearity to the neural network
- $\mathrm{ReLU}(x) = \max(0, x)$




- Other activation functions: sigmoid, tanh, leaky ReLU, ELU, etc
- See `https://pytorch.org/docs/stable/nn.functional.html#non-linear-activation-functions` for more

# Neural networks with multiple hidden layers




- values of hidden units

$$a^{[1]} = \mathrm{ReLU}(W^{[1]}x + b^{[1]})$$
$$a^{[2]} = \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]})$$

- value of output unit
  - regression:
  $$y = W^{[2]}a^{[2]} + b^{[2]}$$
  - classification:
  $$y = \mathrm{Sigmoid}(W^{[2]}a^{[2]} + b^{[2]})$$

# Neural networks with multiple hidden layers

- input layer: $x$
- hidden layers:

$$\begin{aligned} a^{[1]} &= \mathrm{ReLU}(W^{[1]}x + b^{[1]}) \\ a^{[2]} &= \mathrm{ReLU}(W^{[2]}a^{[1]} + b^{[2]}) \\ &\vdots \\ a^{[L]} &= \mathrm{ReLU}(W^{[L]}a^{[L-1]} + b^{[L]}) \end{aligned}$$

- output layer:
  - regression:

$$y = W^{[L+1]}a^{[L]} + b^{[L+1]}$$

  - classification:

$$y = \mathrm{Sigmoid}(W^{[L+1]}a^{[L]} + b^{[L+1]})$$

# Multi-layer perceptrons

- Such neural network architectures are often referred to as **multi-layer perceptrons (MLPs)**.

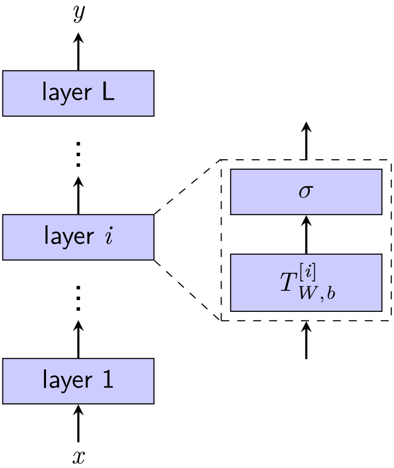
- A layer corresponds to the transformation of

$$z_{\text{out}} = \sigma(Wz_{\text{in}} + b) = \sigma(T_{W,b}(z))$$

  where $\sigma$ is an activation function, $W$ and $b$ are the weight and bias parameters, respectively, and $T_{W,b}$ is the affine transformation $T_{W,b}(z) = Wz + b$.

- Each layer has its own weight and bias parameters.

# Multi-layer perceptrons



# Outline

## Residue connections

- An influential architecture that makes it easier to train networks with many layers
- A residue layer
$$\mathrm{Res}(z) = z + \sigma(T(\sigma(T(z))))$$
- A residue network is a sequence of residue layers
$$\mathrm{ResNet(z)} = T(\mathrm{Res}(\ldots \mathrm{Res}(z)))$$

# MLP and Residue network