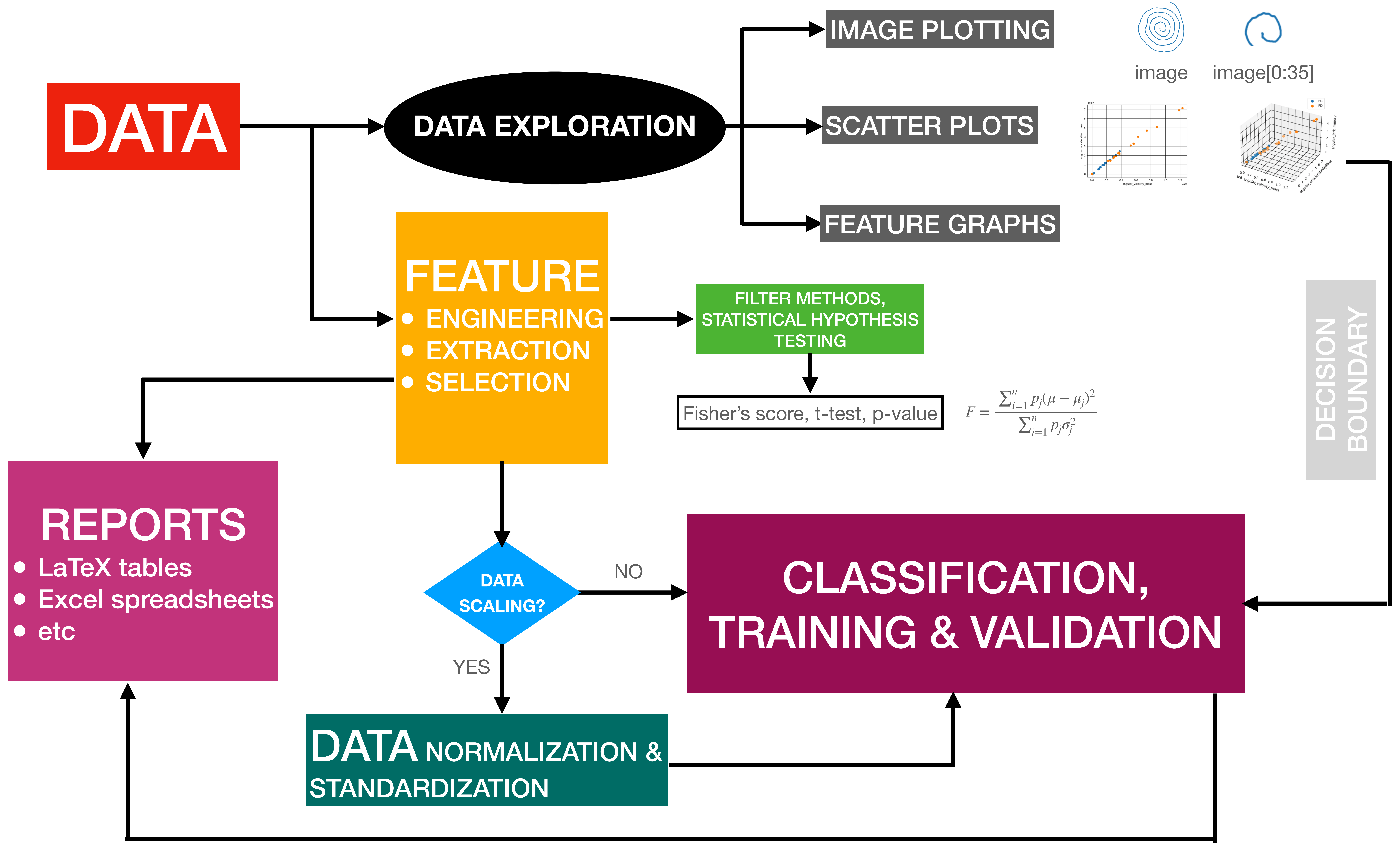
Neural Network Intuition. Backward Propagation. The XOR problem.

Elli Valla

PhD student and junior researcher at

Department of Software Science
TalTech University

elli.valla@taltech.ee


DATA
DATA EXPLORATION
IMAGE PLOTTING
image
image[0:35]
SCATTER PLOTS
FEATURE GRAPHS
FEATURE
ENGINEERING
EXTRACTION
SELECTION
FILTER METHODS, STATISTICAL HYPOTHESIS TESTING
Fisher's score, t-test, p-value
$F = \frac{\sum_{i=1}^{n} p_j(\mu - \mu_j)^2}{\sum_{i=1}^{n} p_j \sigma_j^2}$
DECISION BOUNDARY
REPORTS
LaTeX tables
Excel spreadsheets
etc
DATA SCALING?
NO
YES
CLASSIFICATION, TRAINING & VALIDATION
DATA NORMALIZATION & STANDARDIZATION

model	accuracy_mean	precision_mean	sensitivity_mean	specificity_mean	f1_mean	roc_auc_mean
LogReg	0.8218	0.8133	0.7500	0.8714	0.7683	0.8833
RF	0.8018	0.7367	0.7500	0.8381	0.7341	0.7875
KNN	0.8055	0.8476	0.7000	0.8810	0.7359	0.8792
SVM	0.8418	0.8933	0.7000	0.9381	0.7683	0.8250
DT	0.6655	0.5700	0.6000	0.7095	0.5748	0.6548
AdaBoost	0.7200	0.6667	0.6000	0.8000	0.6286	0.7958
LogReg	0.8218	0.8133	0.7500	0.8714	0.7683	0.8917
RF	0.8018	0.7600	0.7000	0.8714	0.7206	0.8387
KNN	0.8055	0.8476	0.7000	0.8810	0.7359	0.8792
SVM	0.8600	**0.9333**	0.7000	**0.9667**	0.7905	0.8083
DT	0.7636	0.7500	0.7000	0.8048	0.7056	0.7524
AdaBoost	0.7036	0.6167	0.6000	0.7762	0.5957	0.7524
LogReg	0.8218	0.8133	0.7500	0.8714	0.7683	0.8917
RF	0.8018	0.7433	0.7500	0.8381	0.7421	0.8387
KNN	0.8055	0.8476	0.7000	0.8810	0.7359	0.8792
SVM	0.8600	**0.9333**	0.7000	**0.9667**	0.7905	0.9000
DT	0.7636	0.6967	0.7000	0.8048	0.6897	0.7524
AdaBoost	0.6836	0.5833	0.5500	0.7762	0.5481	0.7190
LogReg	**0.8800**	**0.9333**	0.7500	**0.9667**	**0.8286**	**0.9083**
RF	0.7818	0.7600	0.6500	0.8714	0.6921	0.8179
KNN	0.8418	0.8933	0.7000	0.9381	0.7683	0.8125
SVM	0.8418	0.8933	0.7000	0.9381	0.7683	0.8667
DT	0.6655	0.5833	0.6000	0.7095	0.5865	0.6548
AdaBoost	0.7236	0.5833	0.6000	0.8095	0.5767	0.8125

Plan for today:

- A recap of Logistic Regression.
- Logical Operations (OR, AND, NOR, XOR).
- Forward and backward propagation.
- Practice in MATLAB.

Logistic Regression

1 - a cat is in the image

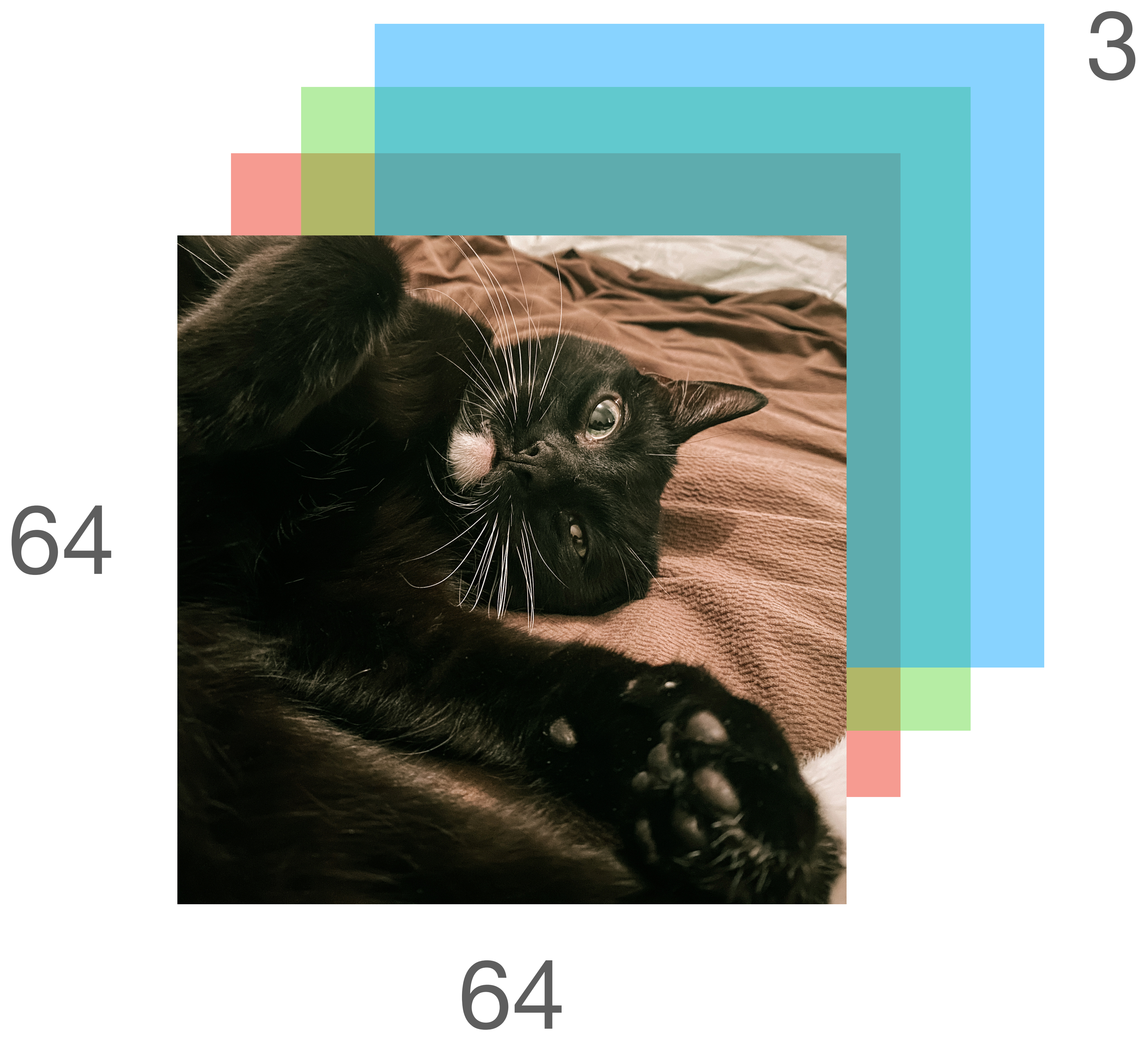
0 - there’s no cat in the image

Logistic Regression

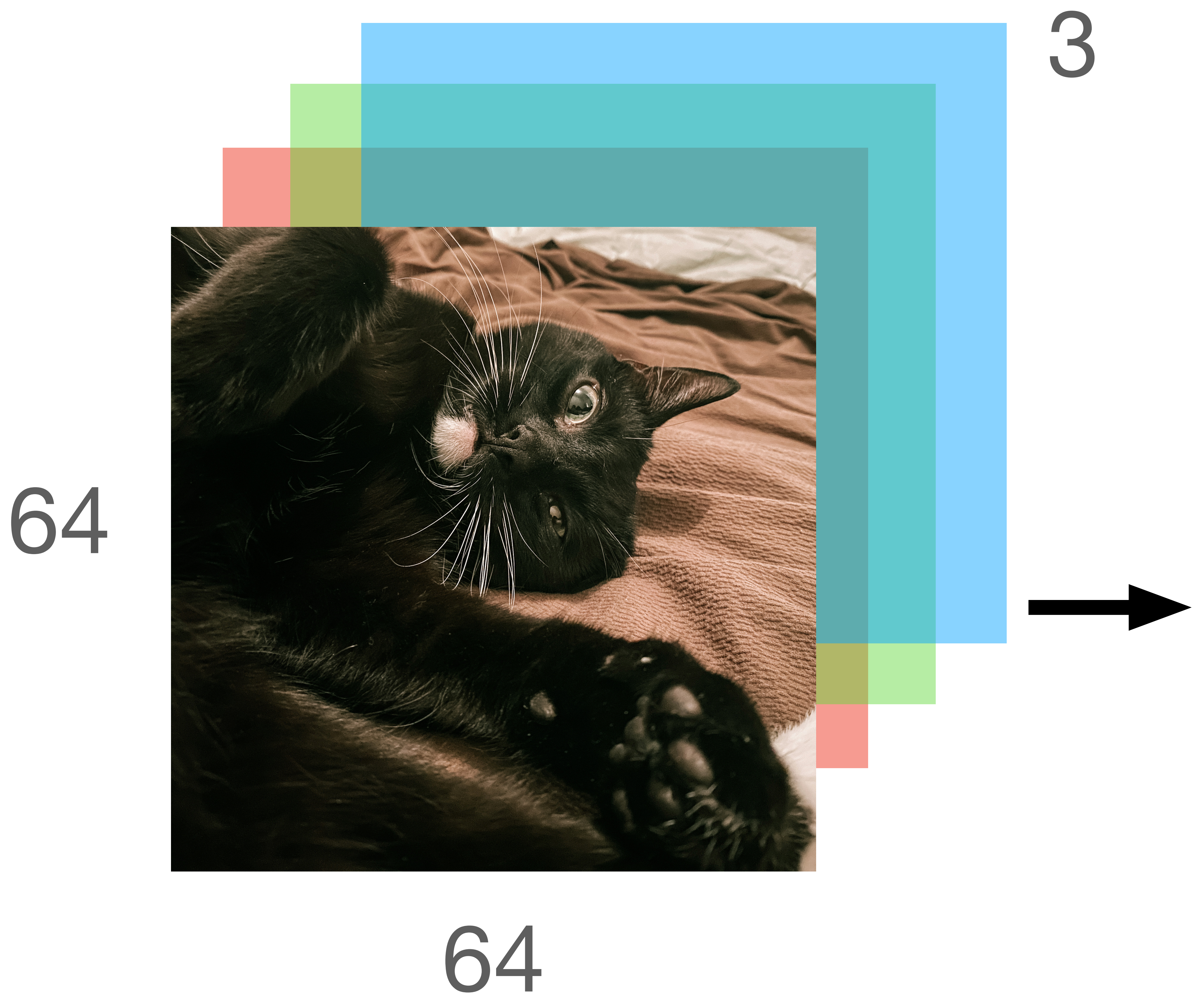
64

64

Logistic Regression

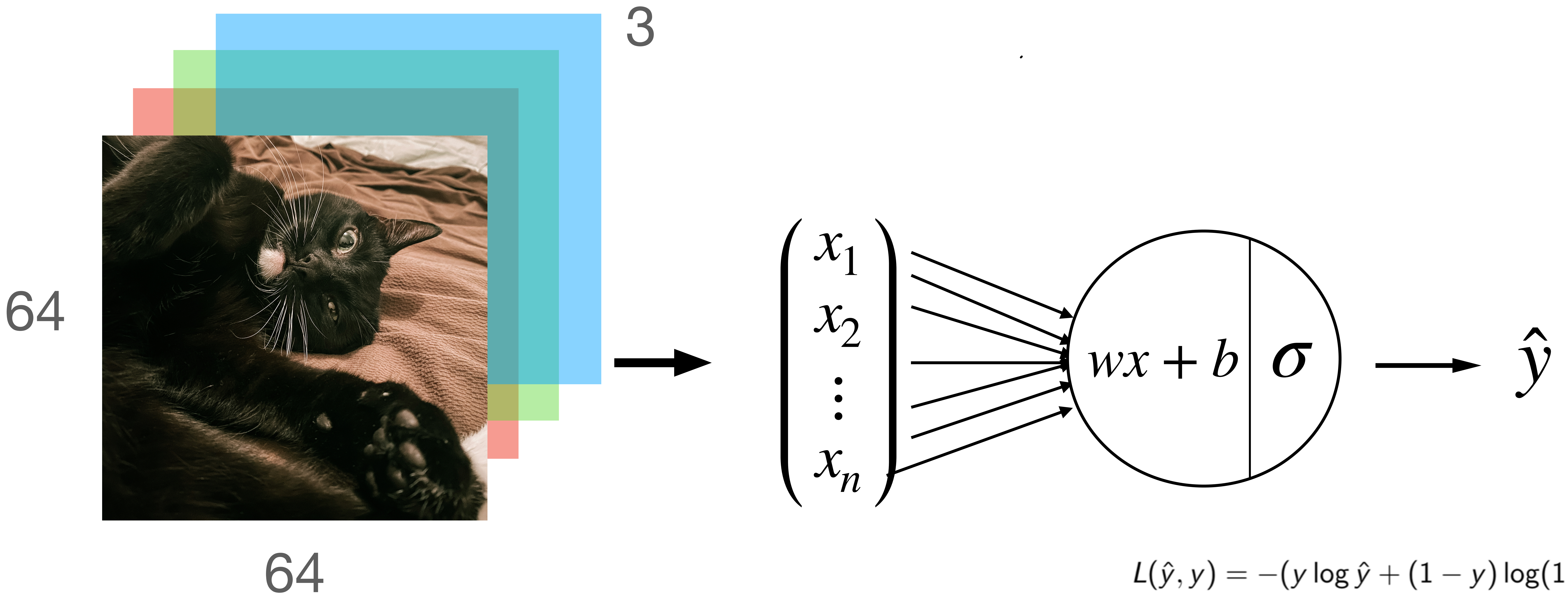


Logistic Regression



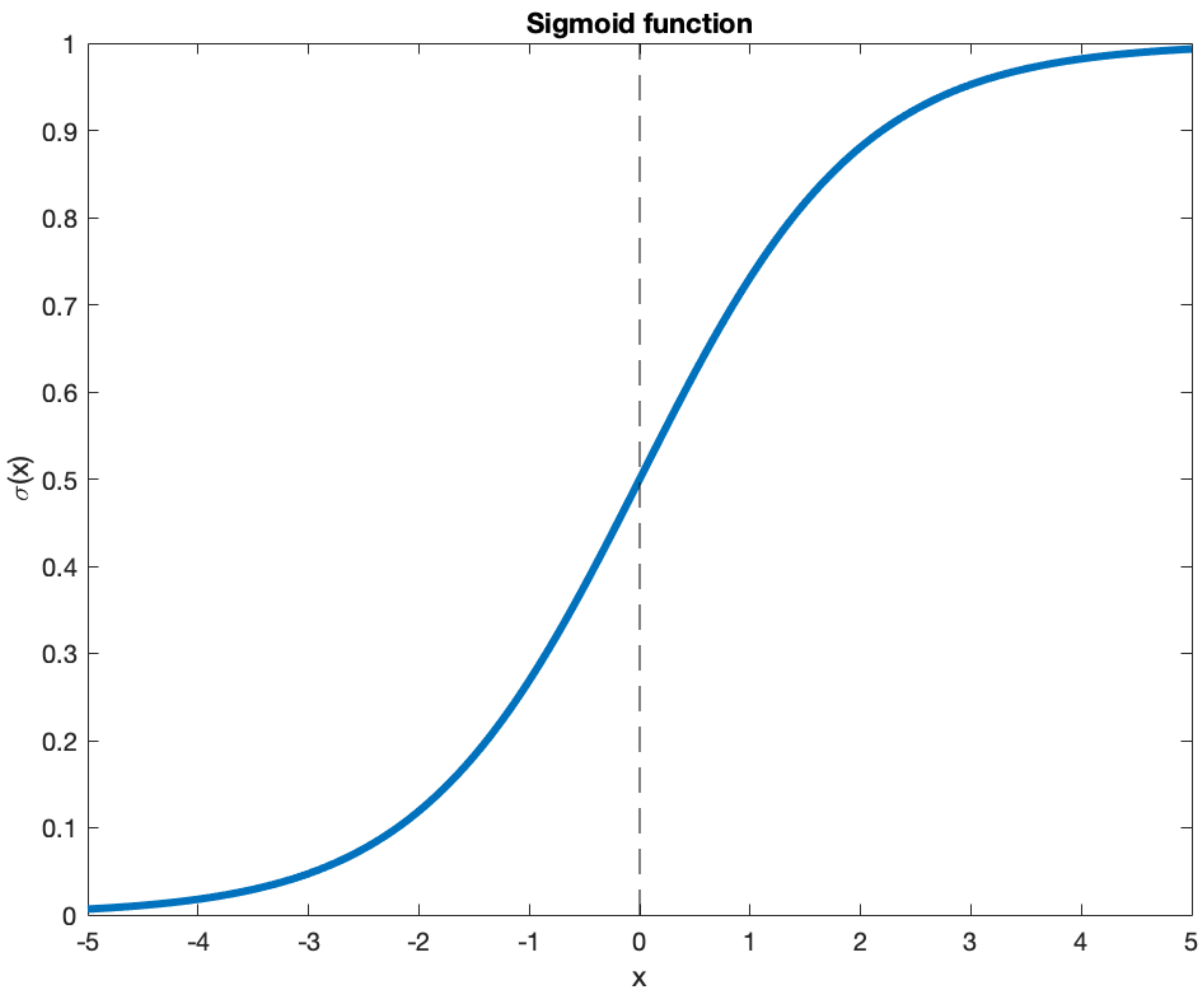
$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Logistic Regression

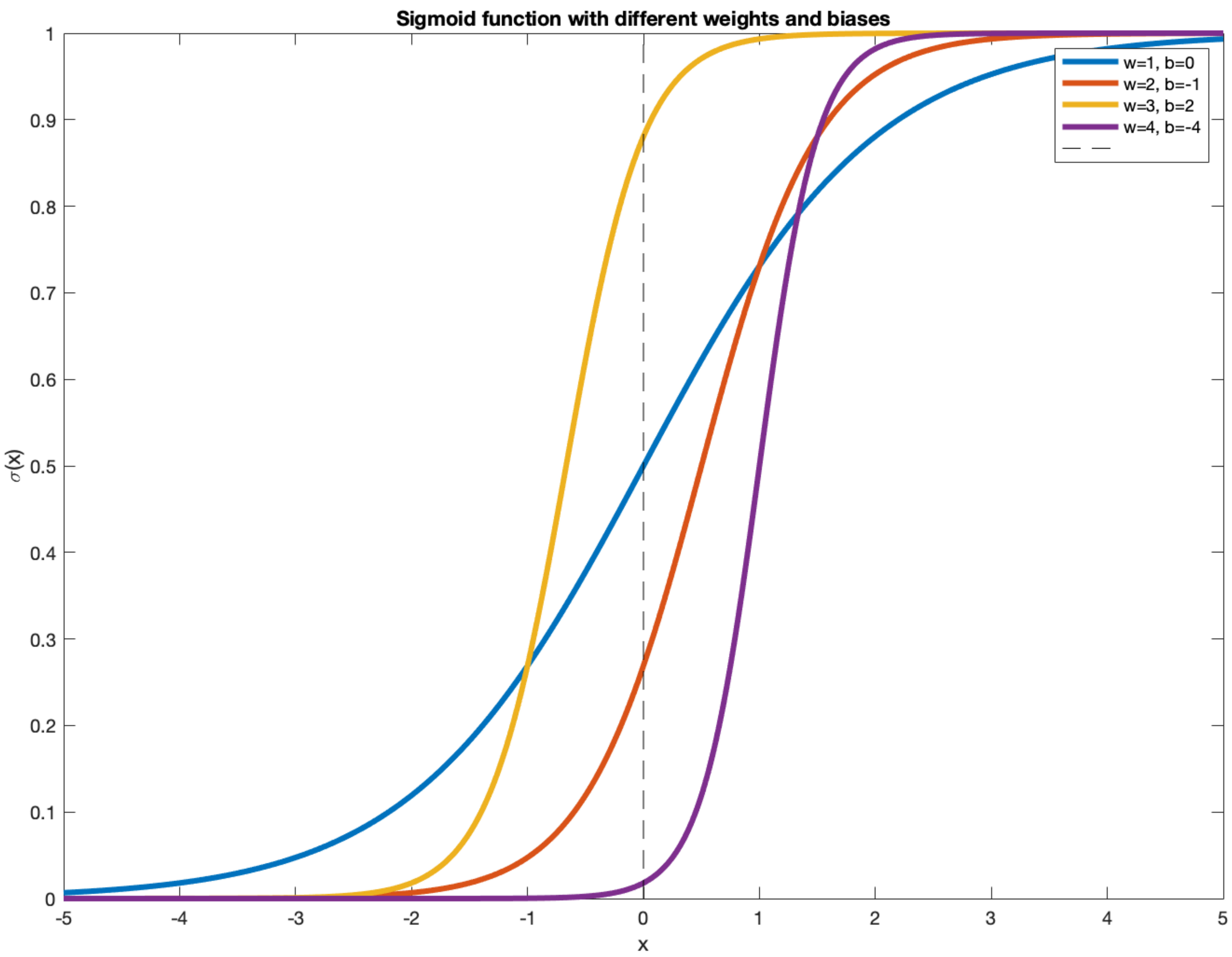


$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

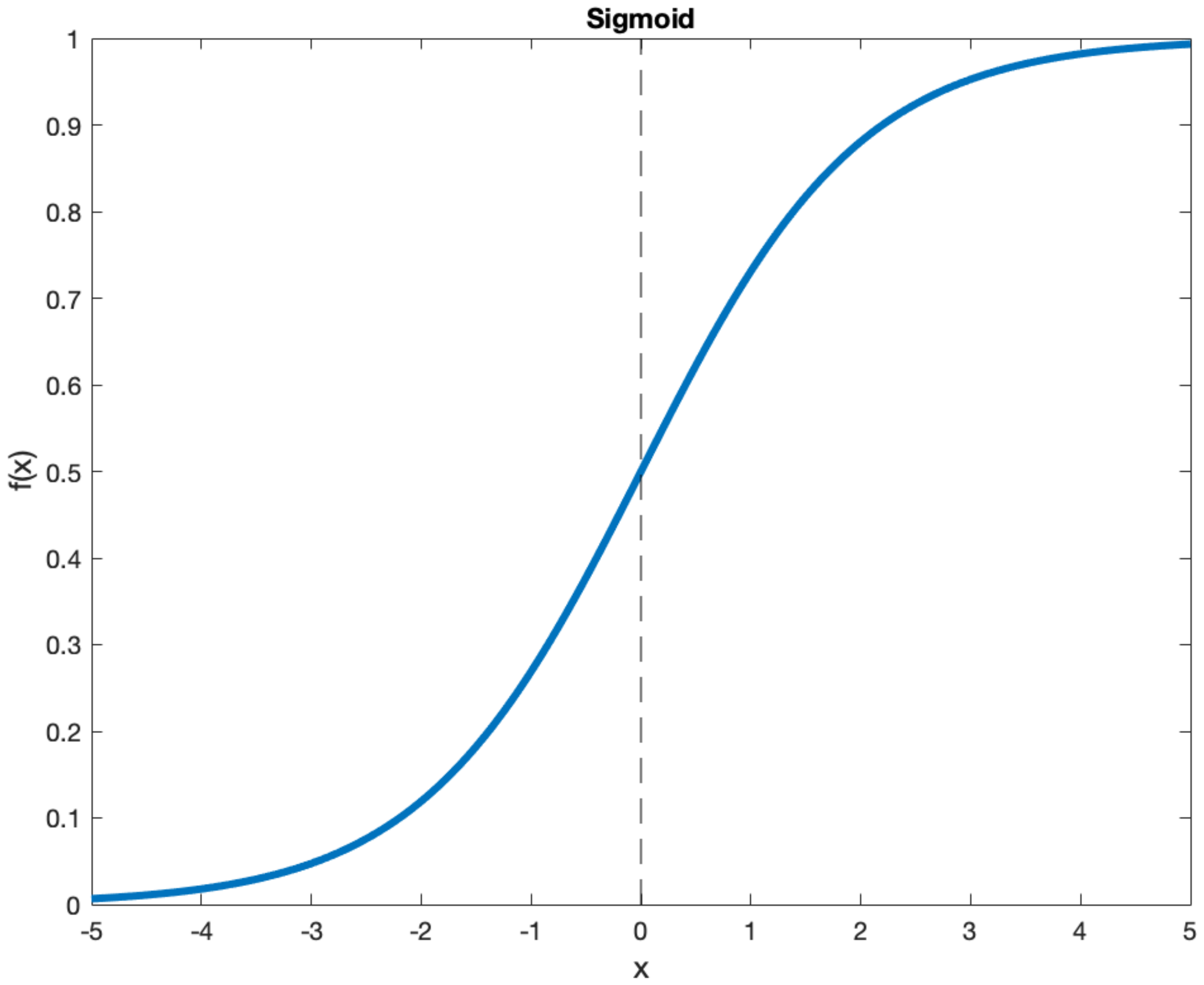
Sigmoid



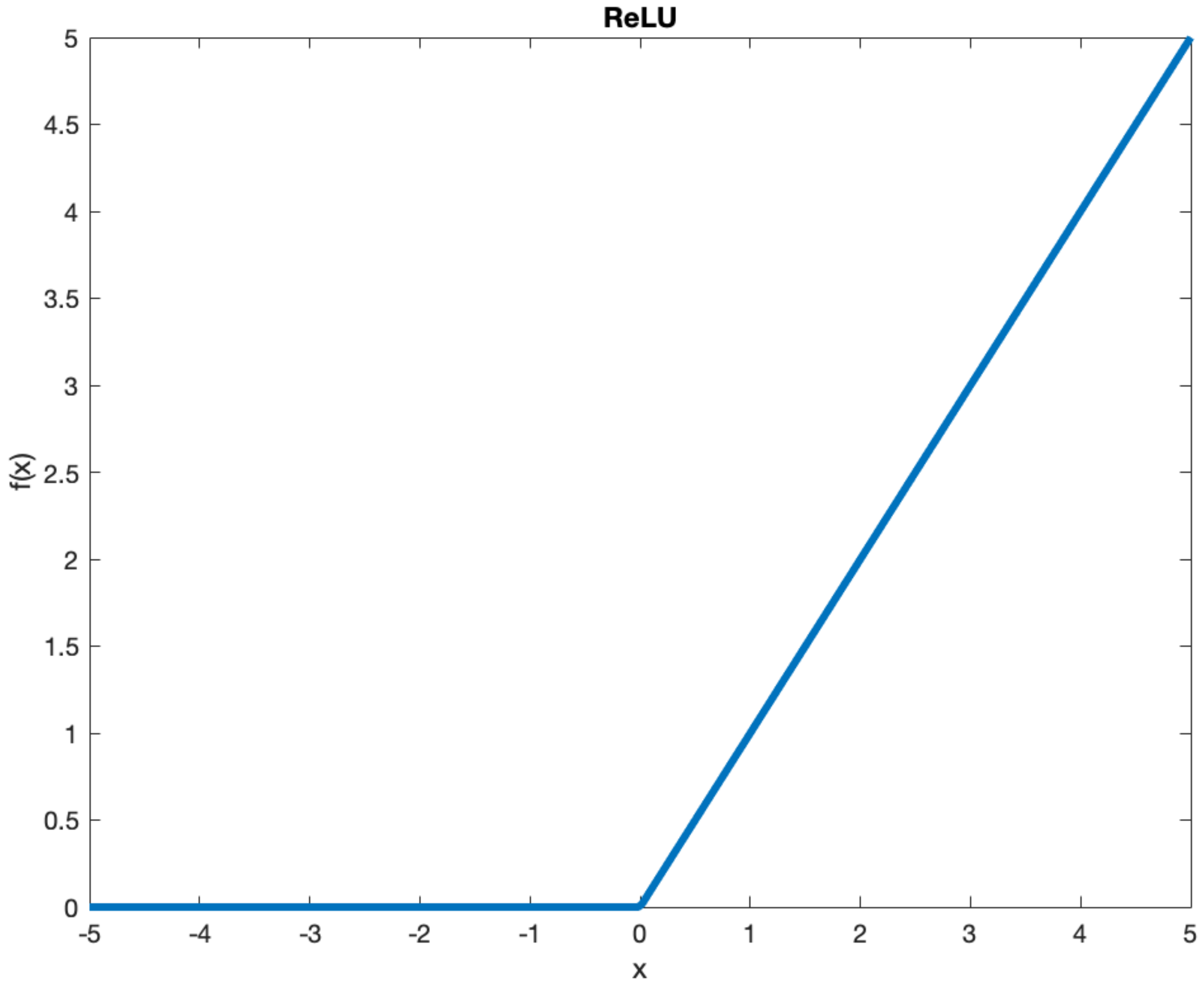
Logistic Regression



Activation functions

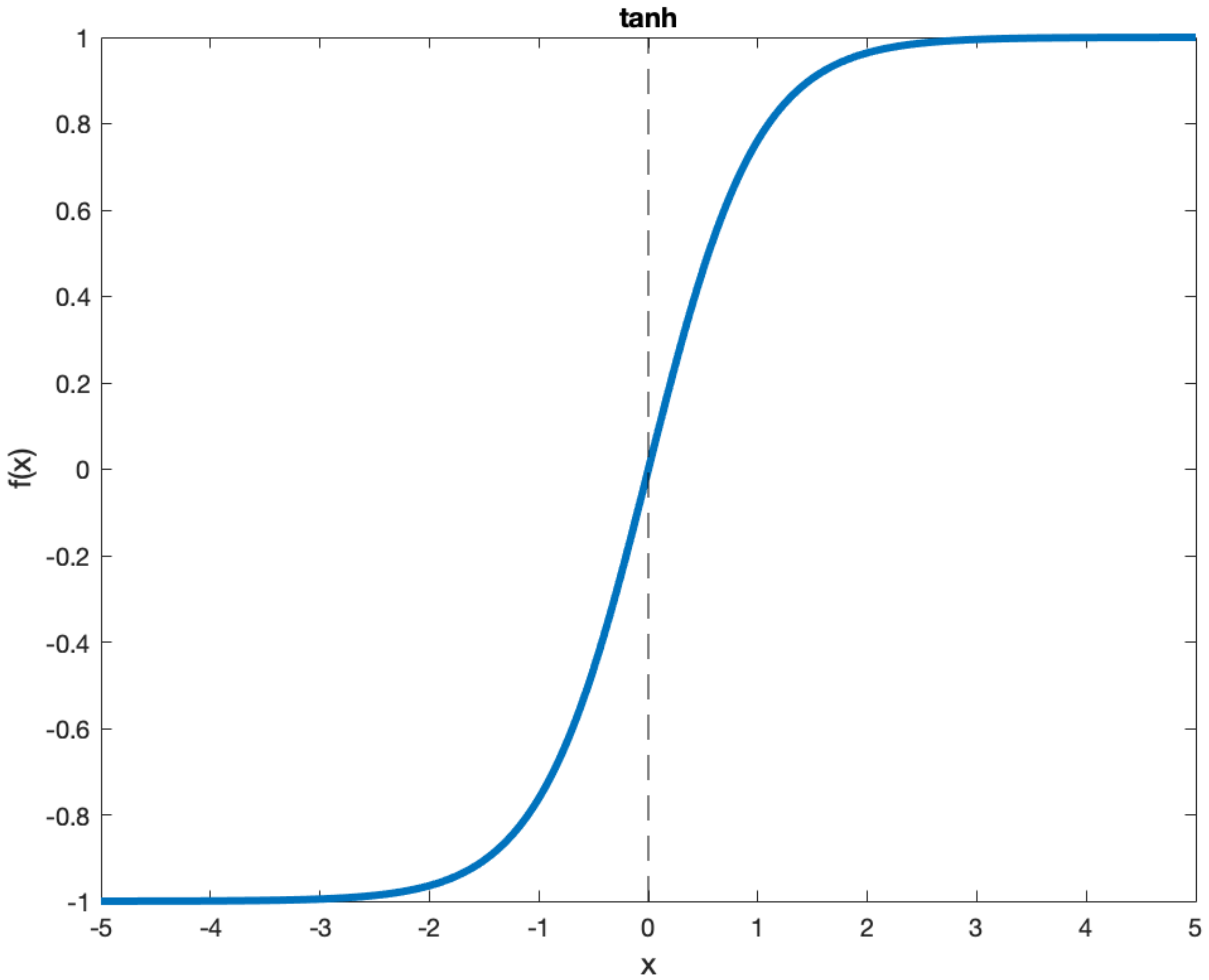


$$\frac{1}{1+e^{-x}}$$



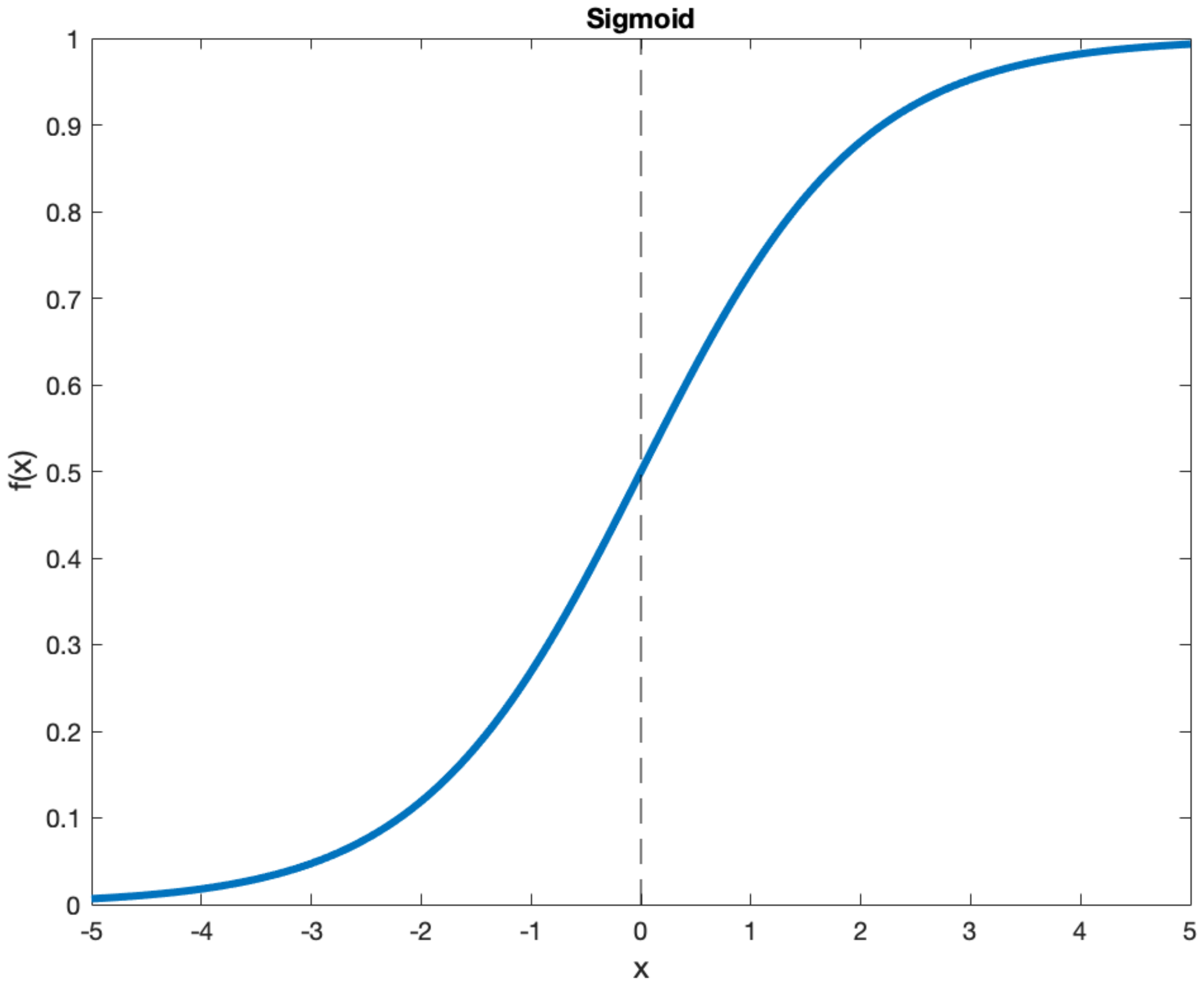
$$\max(0, x)$$

Rectified Linear Unit

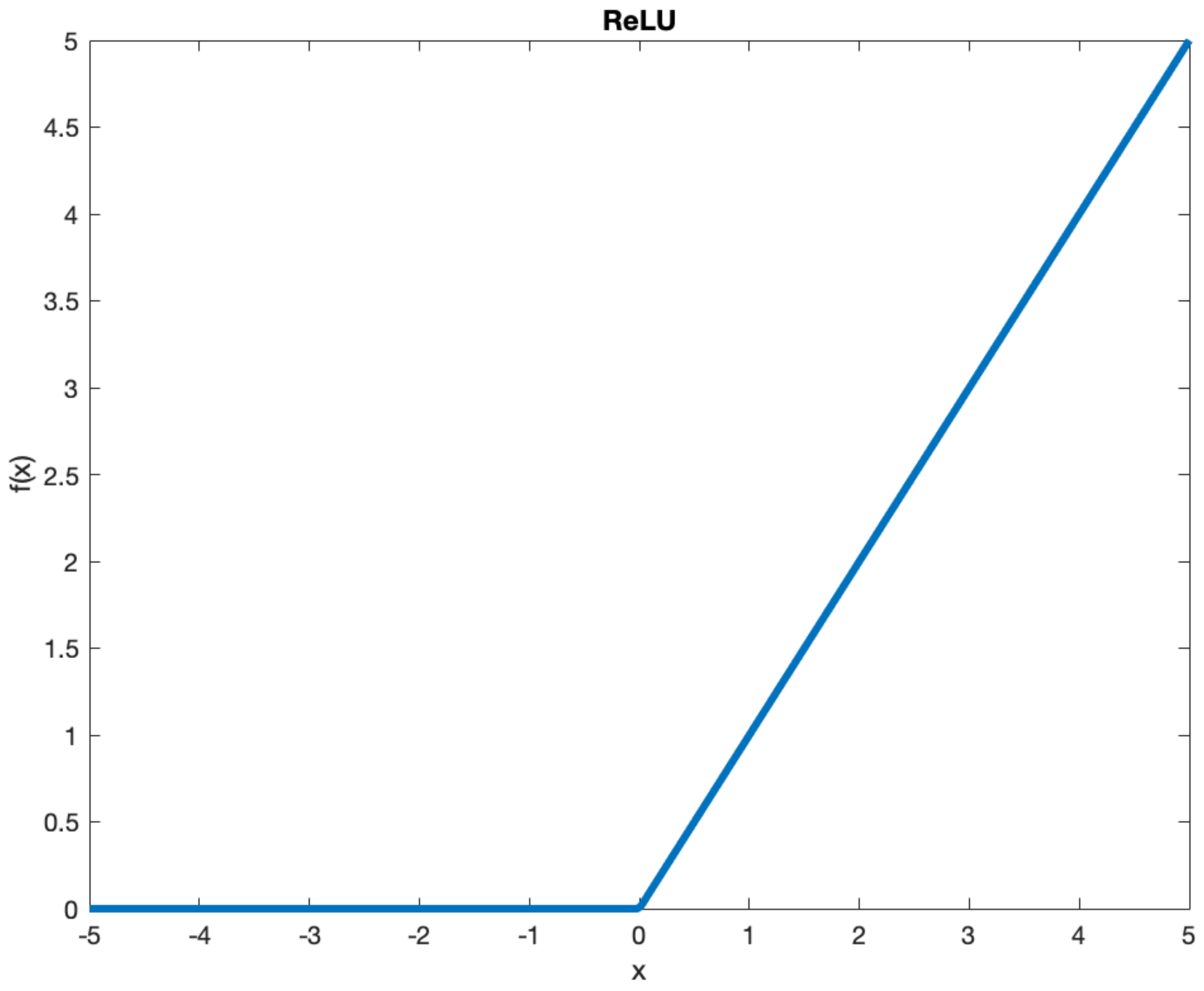


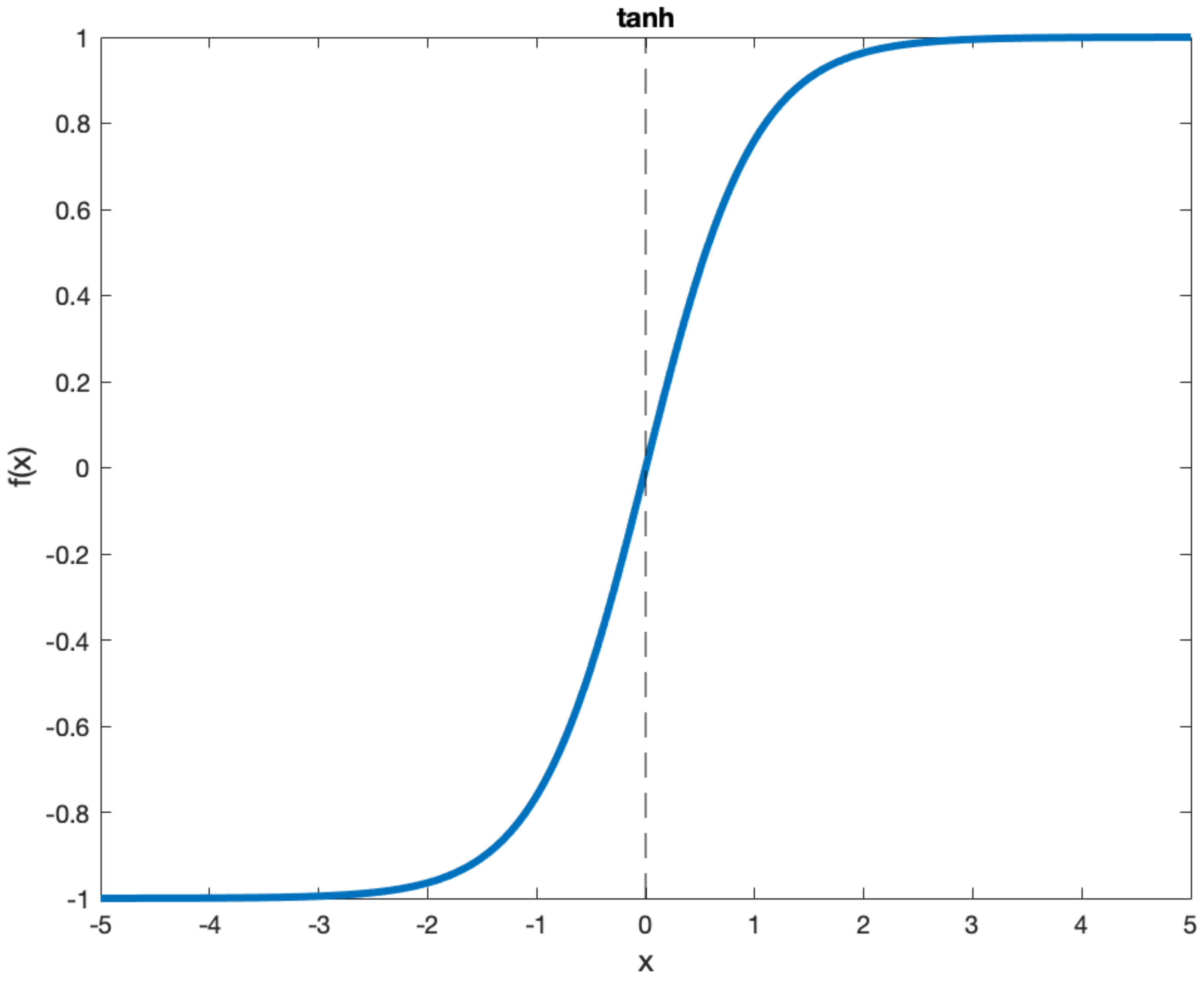
$$\tanh(x)$$

Activation functions



Most common default choice

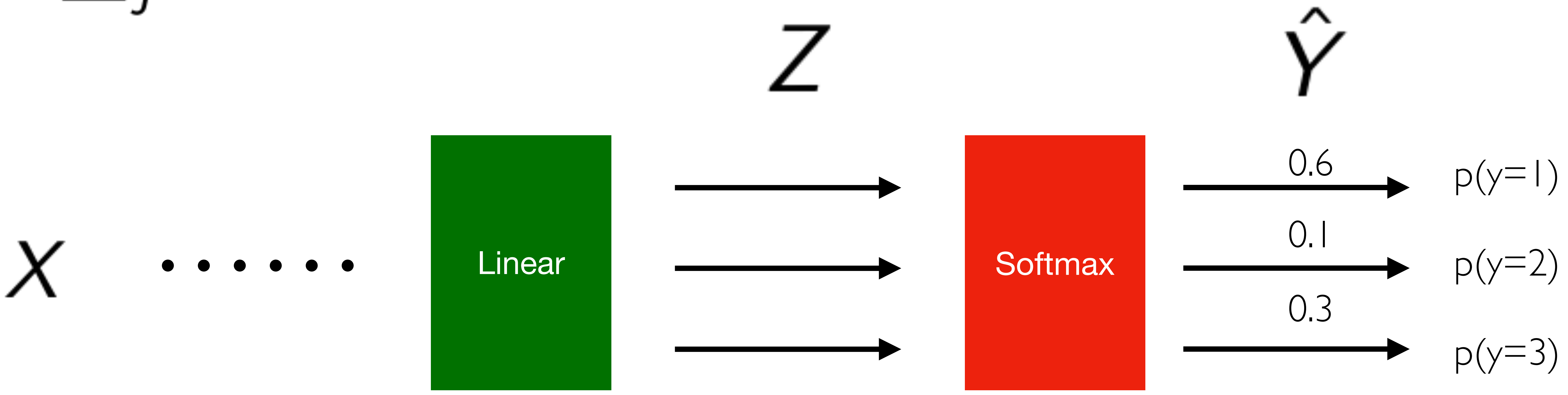




Activation functions

Softmax

$$\sigma(Z)_i = \frac{e^{z_i}}{\sum_j^K e^{z_j}}$$



Training process

1. Initialise parameters.
   - Xavier initialisation[1]
   - He initialisation[2]

```
tf.keras.initializers.GlorotNormal(
    seed=None
)
```

```
tf.keras.initializers.HeUniform(
    seed=None
)
```

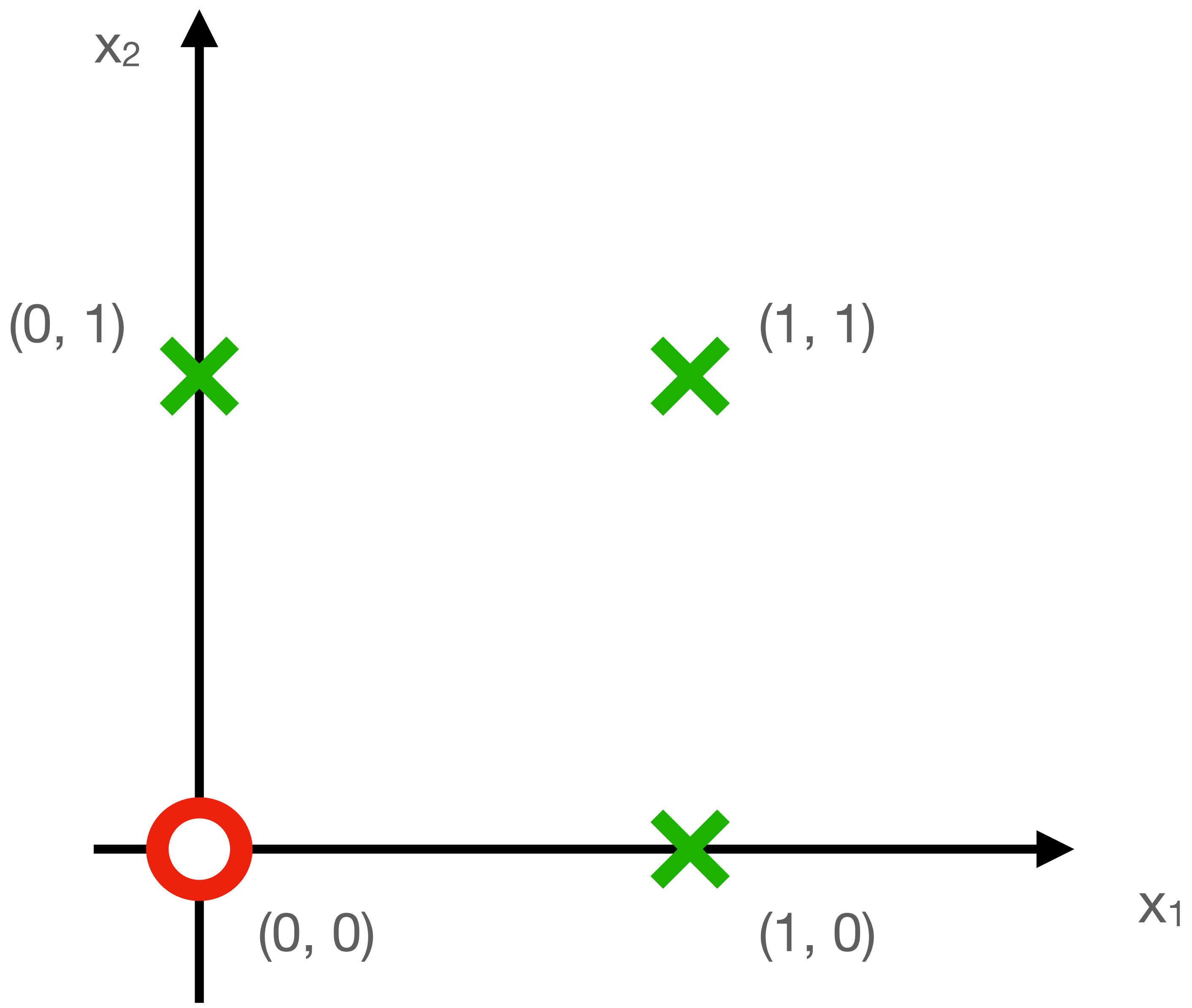
2. Optimise parameters (define the loss function and choose an optimisation algorithm):
   1. Compute the loss function (forward propagation).
   2. Compute the gradients of the loss with respect to parameters (backward propagation).
   3. Update each parameter according to the optimisation algorithm.
3. Use optimised parameters for prediction.
   - Repeat steps 2.1-2.3

[1] Glorot et al, "Understanding the difficulty of training deep feedforward neural networks" (2010)
[2] He et al, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification" (2015)

The OR operation

Inclusive OR

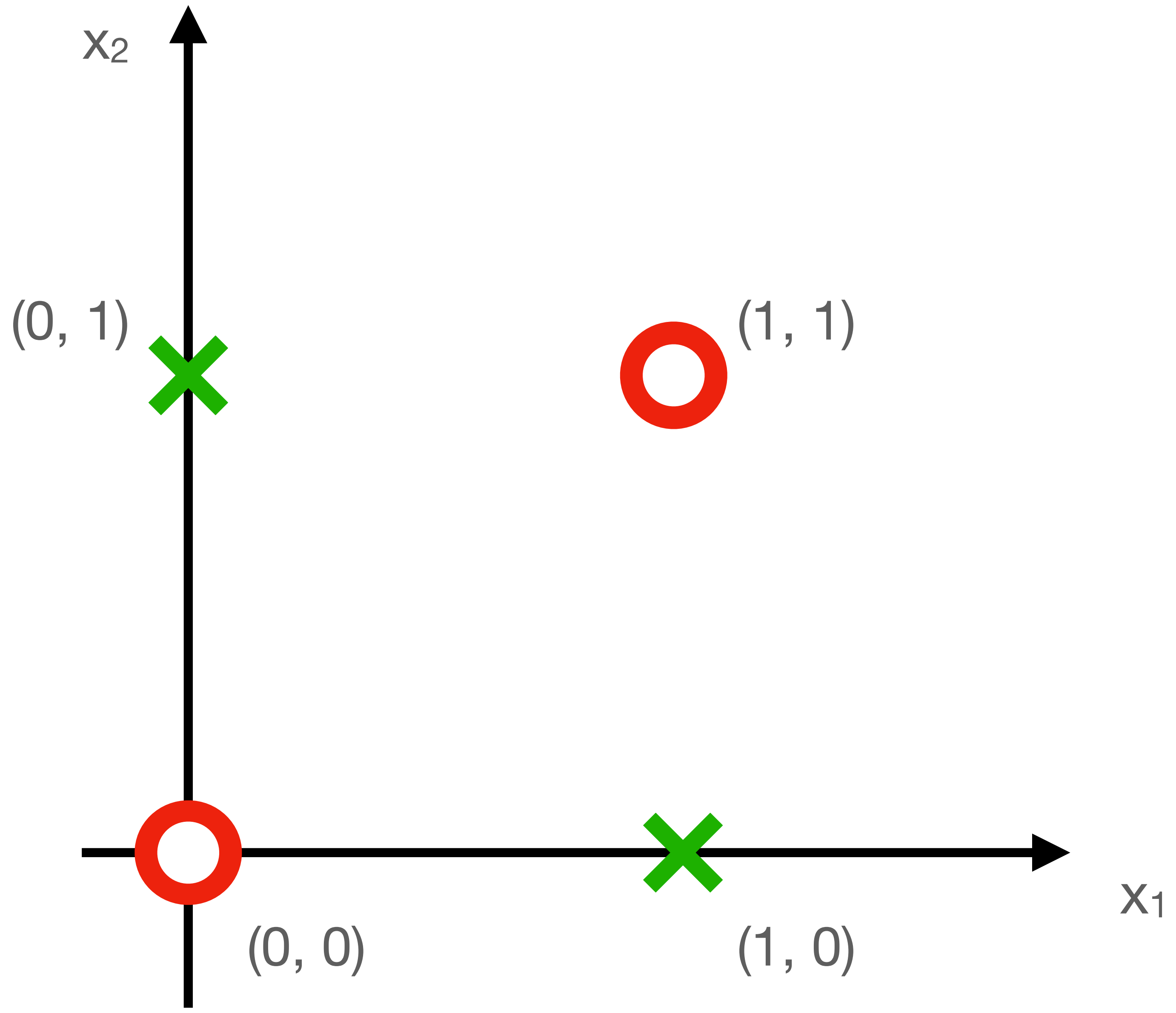
x_1	x_2	target
0	0	0
0	1	1
1	0	1
1	1	1



The XOR operation

Exclusive OR

x_1	x_2	target
0	0	0
0	1	1
1	0	1
1	1	0

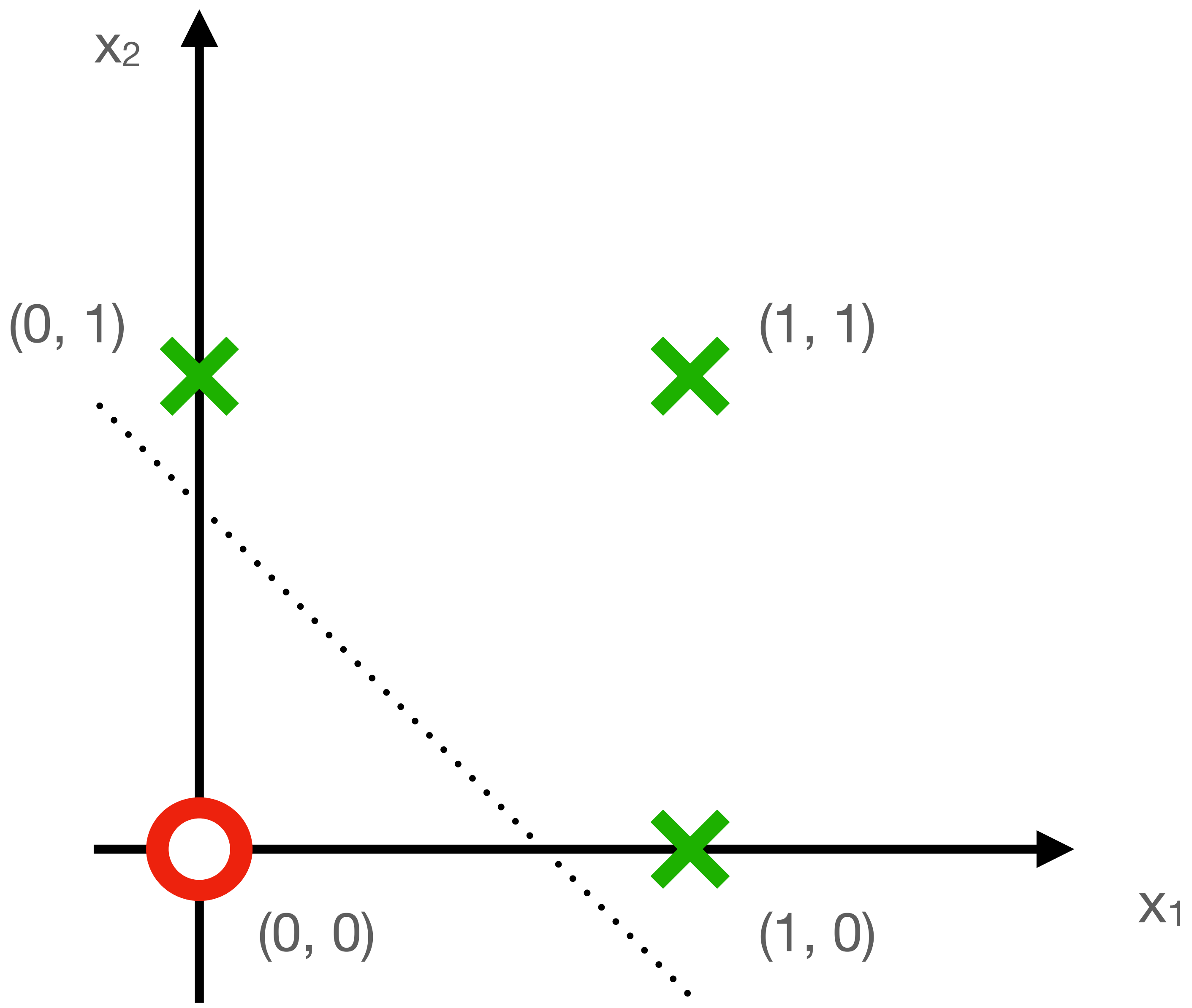


Logical Operators

The OR operation

Inclusive OR

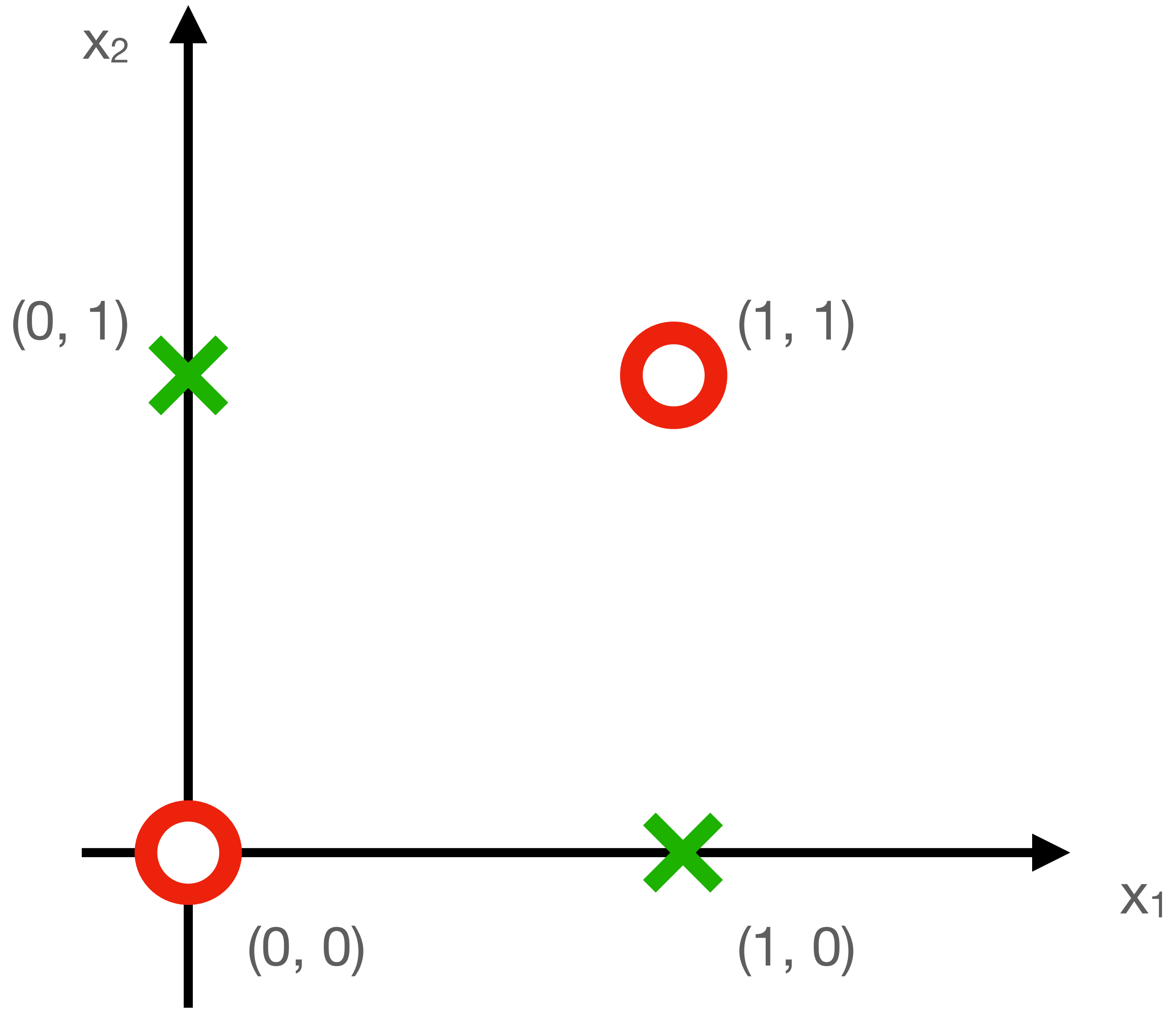
x_1	x_2	target
0	0	0
0	1	1
1	0	1
1	1	1



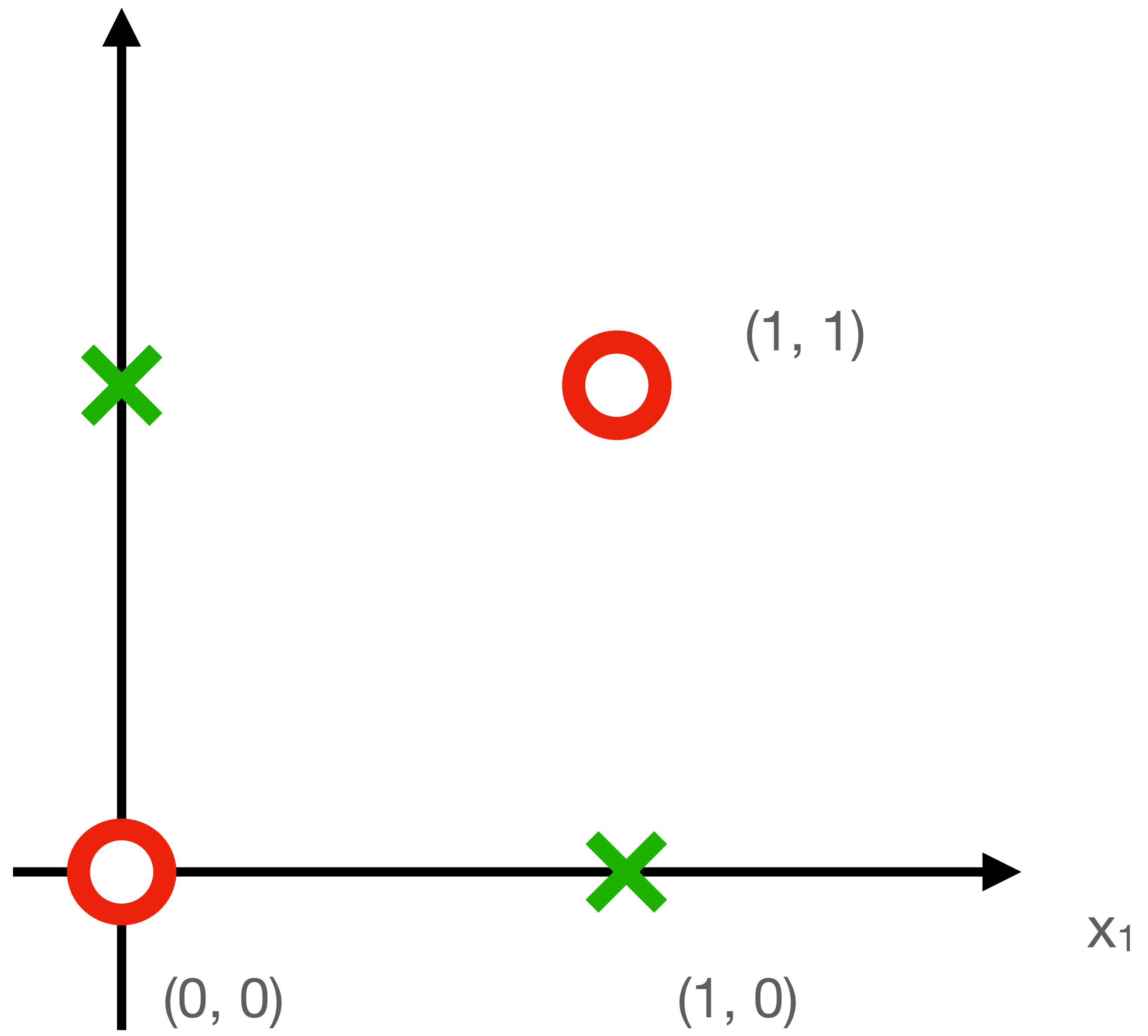
The XOR operation

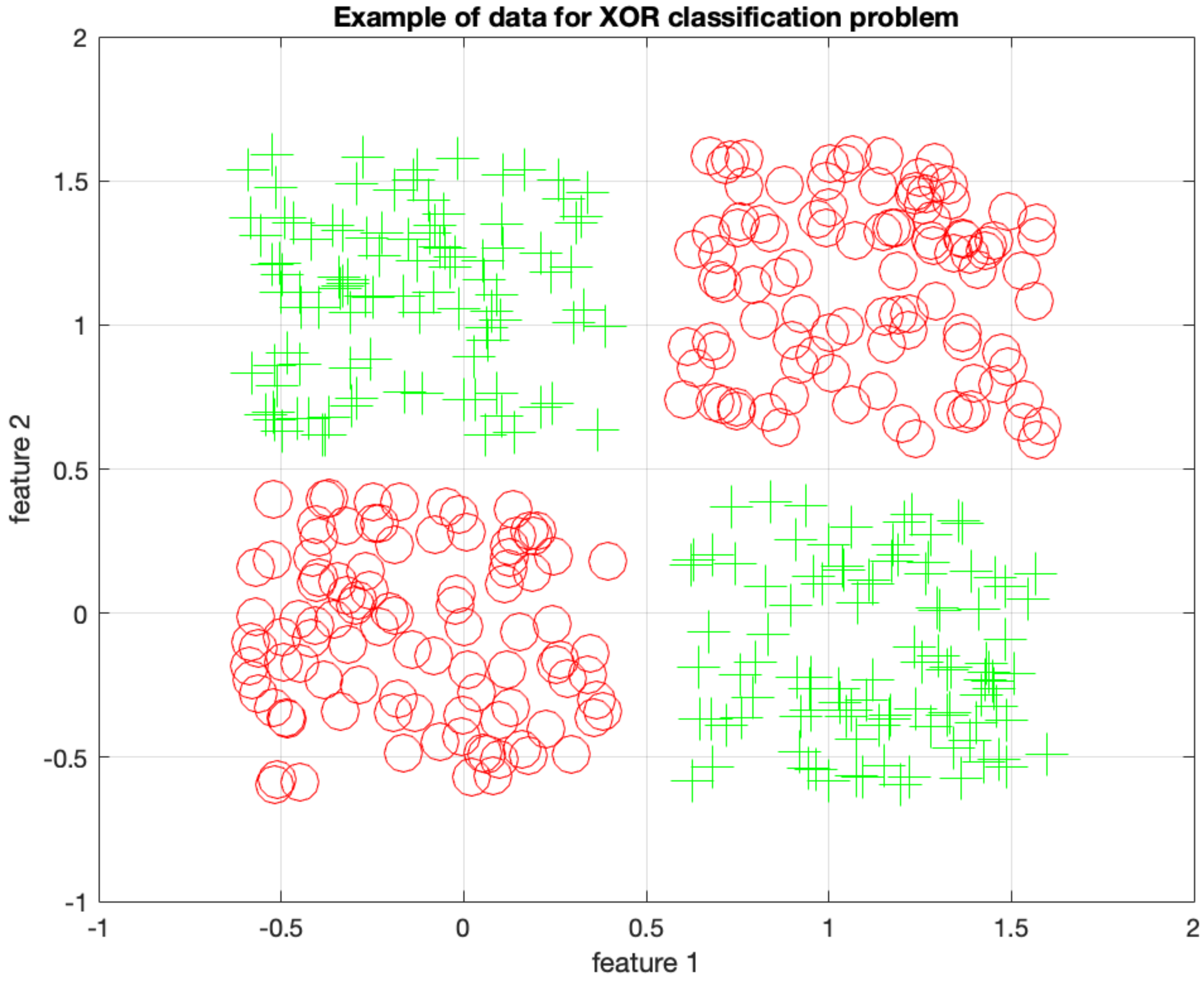
Exclusive OR

x_1	x_2	target
0	0	0
0	1	1
1	0	1
1	1	0

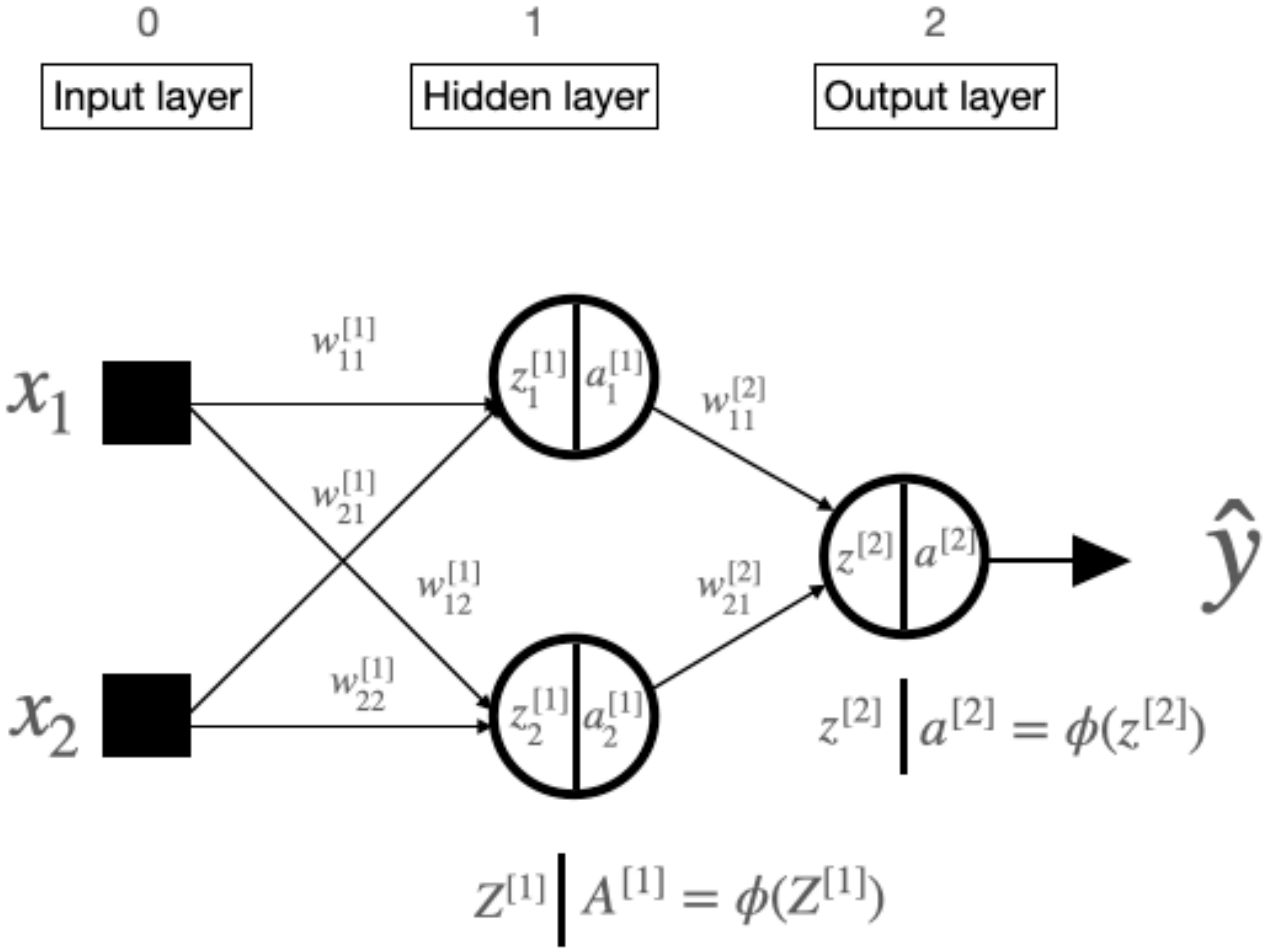


Logical Operators





2-layer neural network



Model Configuration

Activation function

(for both layers)

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Loss function

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

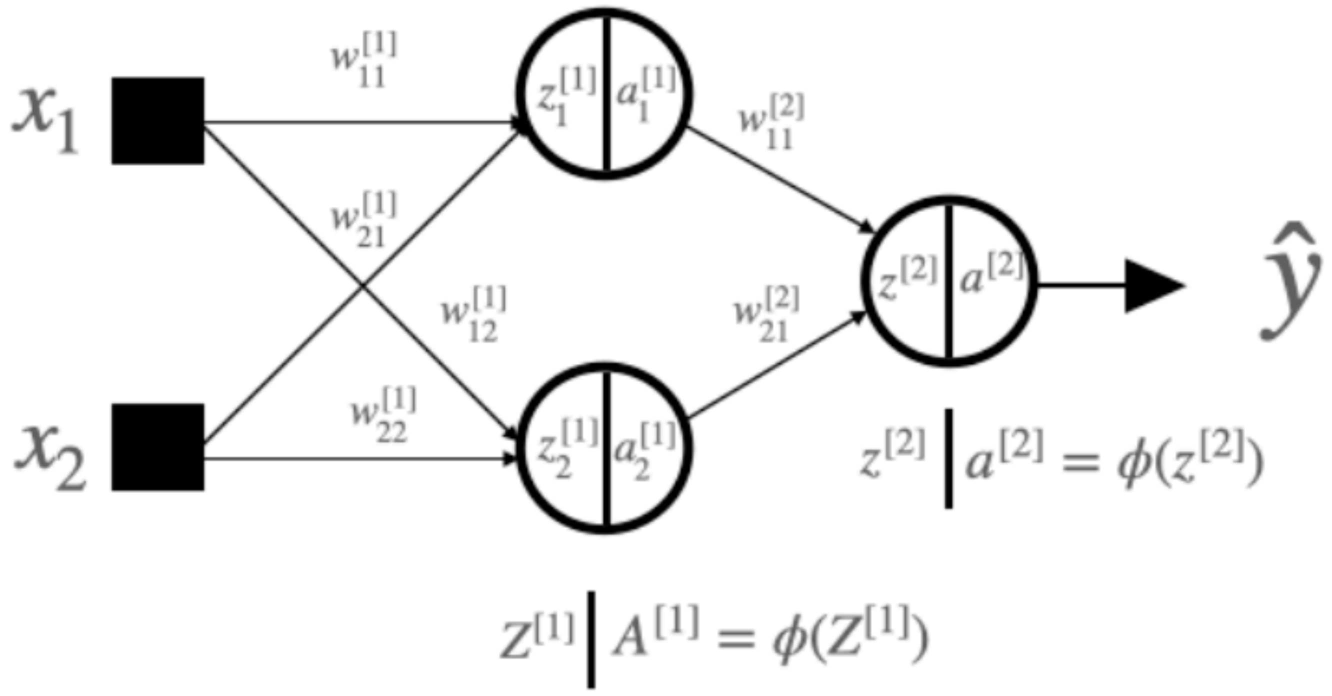
Optimization algorithm

$$Gradient\ decent$$

$$w^{[l]} = w^{[l]} - \alpha \frac{\partial L(y, \hat{y})}{\partial w^{[l]}}$$

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial L(y, \hat{y})}{\partial b^{[l]}}$$

Forward propagation



- 1st layer - linear function:

$$Z^{[1]} = \begin{pmatrix} z_1^{[1]} & z_2^{[1]} \end{pmatrix} = \begin{pmatrix} x_1 w_{11}^{[1]} + x_2 w_{21}^{[1]} & x_1 w_{12}^{[1]} + x_2 w_{22}^{[1]} \end{pmatrix}$$

- 1st layer - activation function:

$$A^{[1]} = \phi(Z^{[1]}) = \begin{pmatrix} a_1^{[1]} & a_2^{[1]} \end{pmatrix} = \begin{pmatrix} \frac{1}{1+e^{-(x_1 w_{11}^{[1]} + x_2 w_{21}^{[1]})}} & \frac{1}{1+e^{-(x_1 w_{12}^{[1]} + x_2 w_{22}^{[1]})}} \end{pmatrix}$$

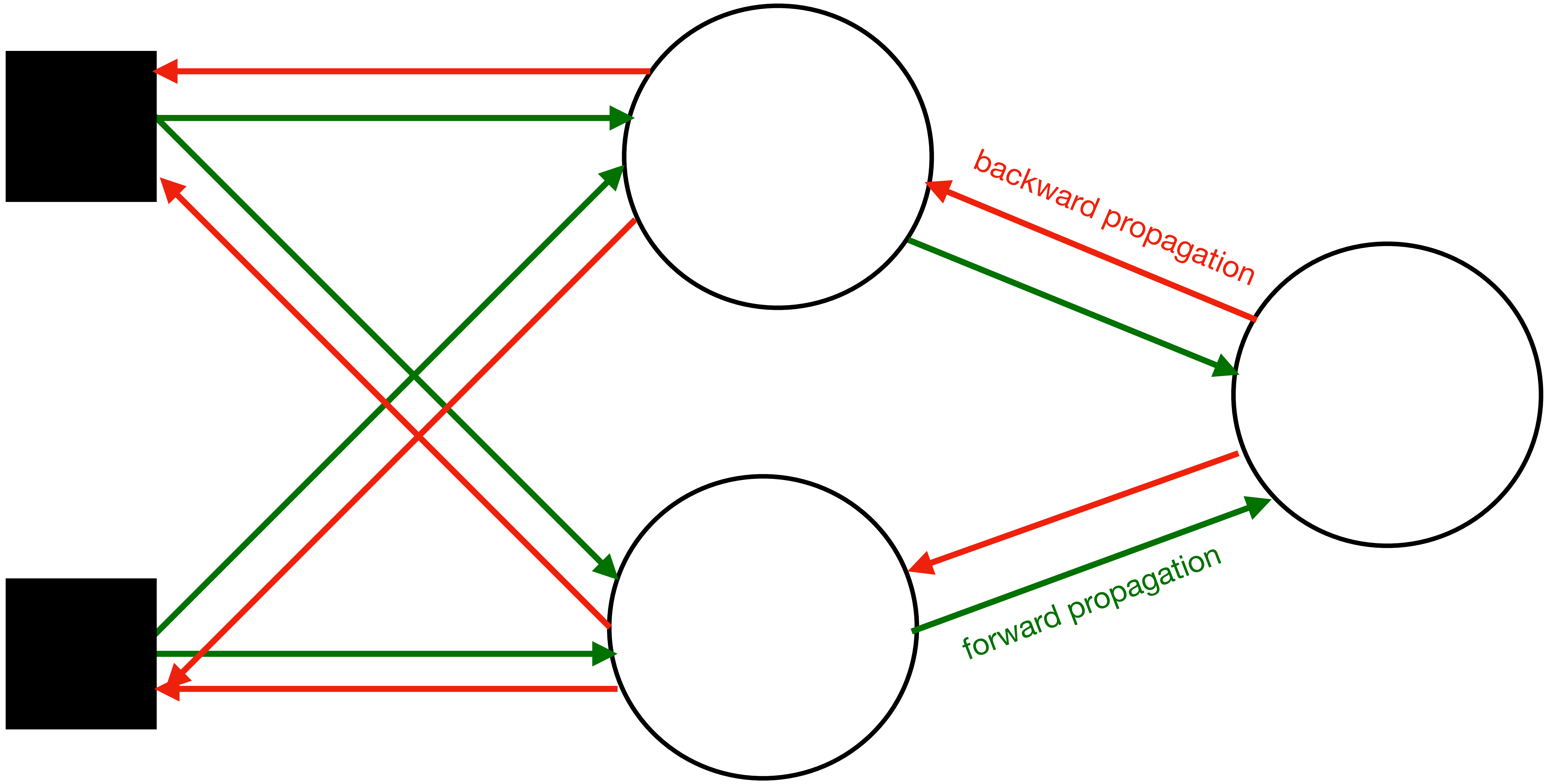
- 2nd layer - linear function:

$$z^{[2]} = a_1^{[1]} w_{11}^{[2]} + a_2^{[1]} w_{21}^{[2]}$$

- 2nd layer activation function:

$$a^{[2]} = \phi(z^{[2]}) = \frac{1}{1 + e^{-(a_1^{[1]} w_{11}^{[2]})}} + \frac{1}{1 + e^{-(a_2^{[1]} w_{21}^{[2]})}}$$

Backward Propagation



Backward Propagation

Sigmoid derivative

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(1-\sigma)$$

$$\frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} = -\frac{y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}} \qquad \frac{\partial a^{[2]}}{\partial z^{[2]}} = a^{[2]} \cdot (1 - a^{[2]})$$

$$\frac{\partial L(a^{[2]}, y)}{\partial z^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} = a^{[2]} - y$$

$$\frac{\partial L(a^{[2]}, y)}{\partial w_{11}^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w_{11}^{[2]}} = (a^{[2]} - y) \cdot a_1^{[1]}$$

$$\frac{\partial L(a^{[2]}, y)}{\partial w_{21}^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w_{21}^{[2]}} = (a^{[2]} - y) \cdot a_2^{[1]}$$

Backward Propagation

Sigmoid derivative

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(1-\sigma)$$

Check:
the gradient with respect to a variable should have the same shape as the variable.

$$\frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} = -\frac{y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}} \qquad \frac{\partial a^{[2]}}{\partial z^{[2]}} = a^{[2]}(1-a^{[2]})$$

$$\frac{\partial L(a^{[2]}, y)}{\partial z^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} = a^{[2]} - y$$

$$\frac{\partial L(a^{[2]}, y)}{\partial z^{[1]}} = \frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} = (a^{[2]} - y) \cdot w^{[2]} \cdot a^{[1]}(1-a^{[1]})$$

$$\frac{\partial L(a^{[2]}, y)}{\partial w^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}} = \frac{\partial L(a^{[2]}, y)}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}} = (a^{[2]} - y) \cdot a^{[1]}$$

$$\frac{\partial L(a^{[2]}, y)}{\partial w^{[1]}} = \frac{\partial L(a^{[2]}, y)}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}} = \frac{\partial L(a^{[2]}, y)}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial w^{[1]}} = (a^{[2]} - y) \cdot w^{[2]} \cdot a^{[1]}(1-a^{[1]}) \cdot x$$

Practice!