MACHINE LEARNING

MA124 Project specifications

University of Warwick

Dr. Matteo Barucco

How to predict rent prices?

#	LOCATION	SIZE OF ROOM (Ft^2)	PEOPLE SHARING	ENSUITE	PRICE PER MONTH (£)
1	EARLSDON	132	2	NO	530
2	KENILWORTH	150	0	YES	970
3	LEAMINGTON	140	1	NO	750
4	KENILWORTH	100	1	YES	720
…	…	…	…	…	…
m	CANLEY	160	2	NO	800

How to predict rent prices?

#	LOCATION	SIZE OF ROOM (Ft^2)	PEOPLE SHARING	ENSUITE	PRICE PER MONTH (£)
1	EARLSDON	132	2	NO	530
2	KENILWORTH	150	0	YES	970
3	LEAMINGTON	140	1	NO	750
4	KENILWORTH	100	1	YES	720
…	…	…	…	…	…
m	CANLEY	160	2	NO	800
	x_0	x_1	x_2	x_3	y

FEATURES

LABELS

Function to learn

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \xrightarrow{\textit{Unknown function}} y$$

GOAL: learn this function

<u>HOW?</u>

1. Use the data you have collected to ***"train"*** your model to recognise patterns in the data.
2. Once model is trained, ***"test"*** it on new unseen data and try to predict rent of new houses.

<u>REQUIRES:</u> A <u>**LOT**</u> of data (10K, 100K, 1000K examples)

Types of learning

SUPERVISED LEARNING:

The model learns from being given “right answers” (labels).

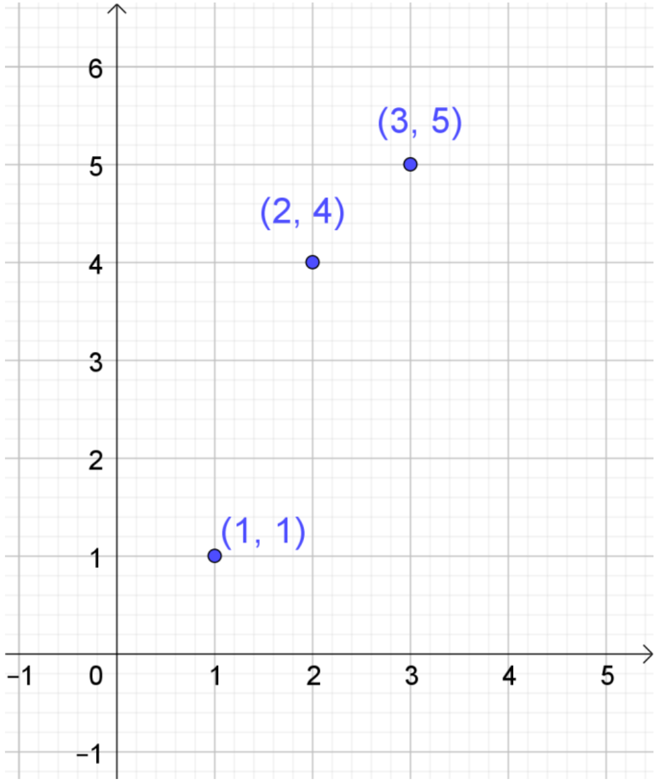
- Predict house rent/house price. (Regression problems)
- Classify an image as “cat” or “dog”. (Classification problems)
- …

UNSUPERVISED LEARNING:

The model does not have “right answers” (labels).

- Generate new music.
- Generate a new image.
- Fly a helicopter.
- …

How does the model learn? TASK A2



- 3 points in the plane.

$$1 \longrightarrow 1$$
$$2 \longrightarrow 4$$
$$3 \longrightarrow 5$$

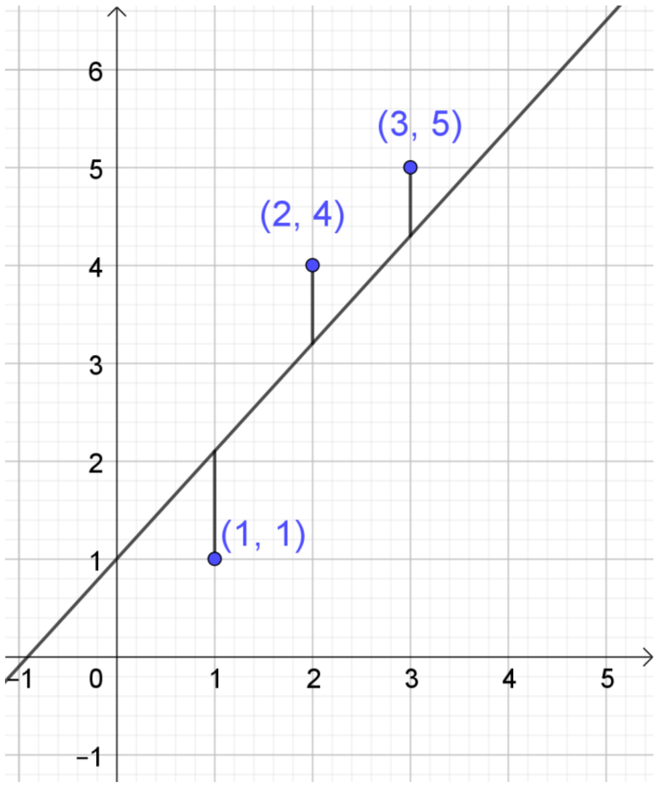
x	y
1	1
2	4
3	5

FEATURE LABEL

- GOAL: find the line of best fit
- MODEL: $y = mx + c$

PARAMETERS TO LEARN

How does the model learn? TASK A2



- Start with random values m_0 and c_0.
- Initial predictions:

$$\hat{y} = m_0 x + c_0$$

- <u>LOSS:</u> For each point, squared distance

$$\hat{y} \longleftrightarrow y$$

- <u>COST:</u> Average of these 3 losses.
- <u>GOAL:</u> find m and c that minimize the COST.

Optimization: Gradient descent.

<u>GOAL:</u> find m and c that minimize the cost $J(m,c)$.

- Start in $J(m_0, c_0)$
- Compute the direction of maximum descent : $-(\frac{dJ}{dm}, \frac{dJ}{dc})$
- Take a step of length α towards maximum descent:

$$(m_1, c_1) = (m_0, c_0) - \alpha \cdot (\frac{dJ}{dm}, \frac{dJ}{dc})$$

- Repeat from (m_1, c_1) until you reach the minimum.

Binary classification: Cat or dog?

#	WEIGHT (Kg)	HEIGHT(cm)	1 = Cat 0 = Dog
1	4.23	23	1
2	3.13	15	0
3	5.32	105	0
4	4.47	25	1
…	…	…	…
m	5.31	110	0

Binary classification: Cat or dog?

#	WEIGHT (Kg)	HEIGHT(cm)	1 = Cat 0 = Dog
1	4.23	23	1
2	3.13	15	0
3	5.32	105	0
4	4.47	25	1
…	…	…	…
m	5.31	110	0
	x_0	x_1	y

FEATURES **LABELS**

Binary classification

For each training example we have:

- x : feature vector containing the features of the animal $\quad x = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$
- y : label containing 0 for dog or 1 for cat.

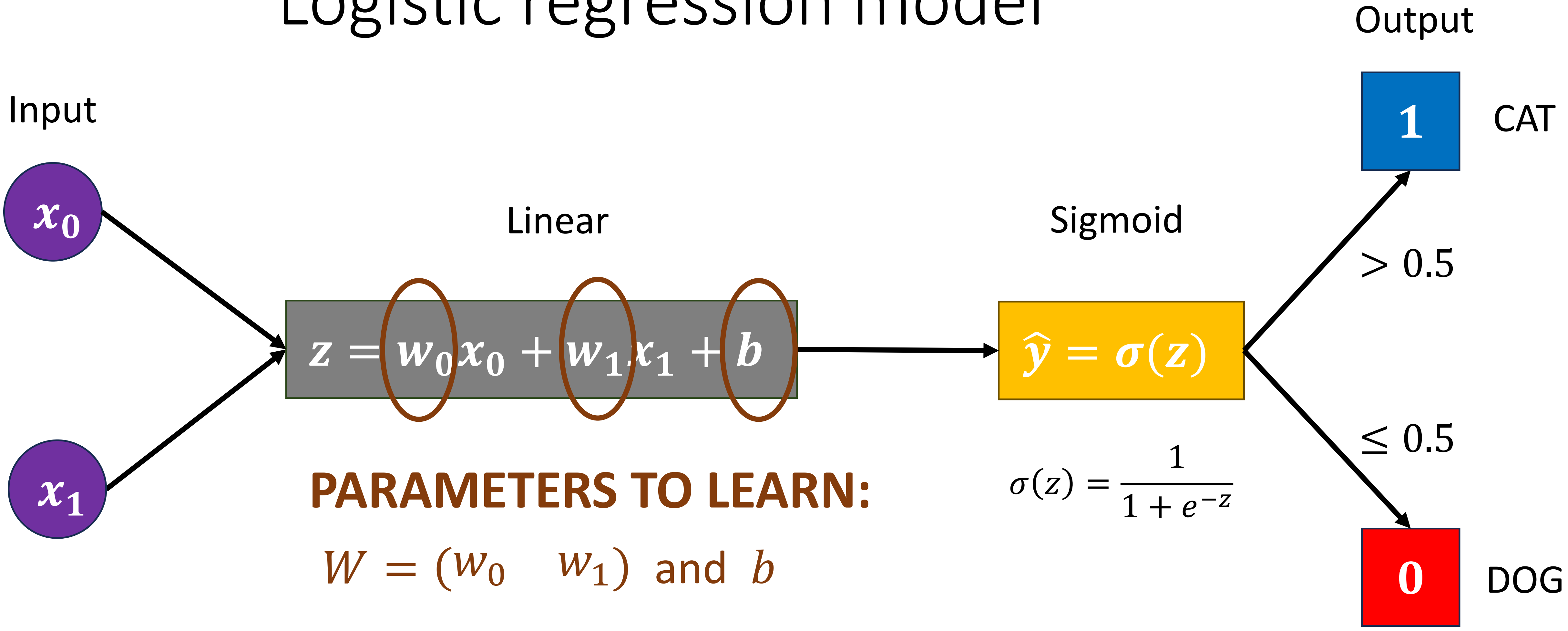
<u>Model:</u>

- Input: features x
- Output: $\hat{y}$, the probability it is either a cat or a dog.

If $\hat{y} > 0.5$ it is a cat

If $\hat{y} \leq 0.5$ it is a dog

Logistic regression model



PARAMETERS TO LEARN:

$W = (w_0 \quad w_1)$ and b

Binary Cross-entropy Loss

For binary classification problems we use the following loss function:

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

When $y = 1$:

- $\hat{y} \sim 0 \quad L(\hat{y}, 1) \sim \infty$
- $\hat{y} \sim 1 \quad L(\hat{y}, 1) \sim 0$

When $y = 0$:

- $\hat{y} \sim 0 \quad L(\hat{y}, 0) \sim 0$
- $\hat{y} \sim 1 \quad L(\hat{y}, 0) \sim \infty$

Cost function

For every training example $i = 1, \ldots, m$

- $x^{(i)}$: Features vector.
- $y^{(i)}$: Label
- $\hat{y}^{(i)} = \sigma(W \cdot x^{(i)} + b)$: Prediction of the model

<u>COST</u> : average of the losses on all training examples

$$J(W, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

<u>GOAL</u> : find W and b that minimize the total cost. $\min_{W,b} J(W, b)$

Optimization: Gradient descent

- Step 0 : Initialize W, b randomly.
- Step 1 : Compute cost $J(W, b)$.
- Step 2 : Compute derivatives of cost with respect to parameters W, b .
- Step 3 : Update parameters with a step α towards minimum:

$$w_0 = w_0 - \alpha \cdot \frac{\partial J}{\partial w_0}$$

$$w_1 = w_1 - \alpha \cdot \frac{\partial J}{\partial w_1}$$

$$b = b - \alpha \cdot \frac{\partial J}{\partial b}$$

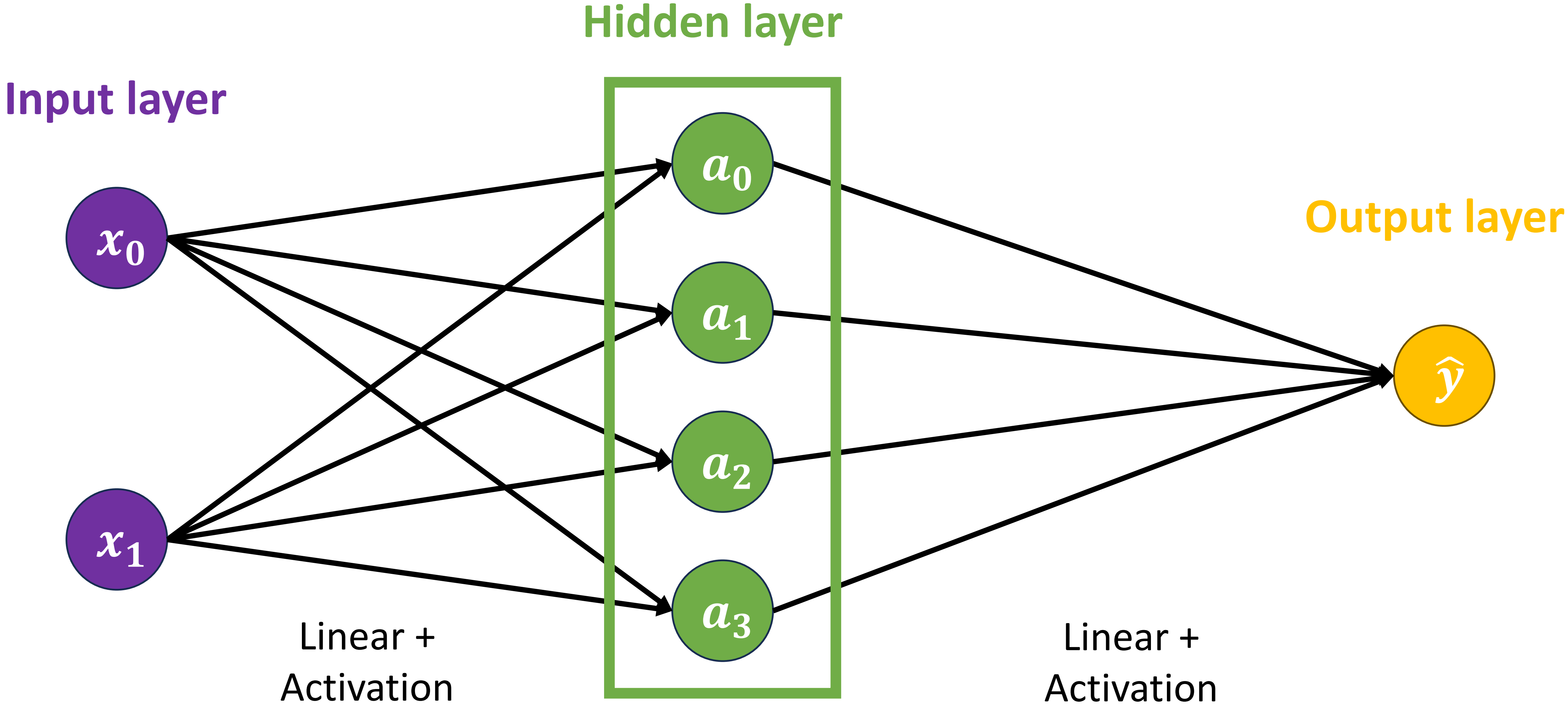
- Repeat steps 1 to 3 till you reach minimum.

Logistic Regression Sum Up

- Initialize model parameters: W, b.
- Train the model parameters (training loop):
    1. Forward propagation: $\hat{y}^{(i)} = \sigma(W \cdot x^{(i)} + b)$
    2. Compute cost: $J = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$
    3. Backward propagation: $\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial b}$
    4. Update parameters: $b = b - \alpha \cdot \frac{\partial J}{\partial b}$
- With the trained W and b predict on new data (test phase):

  If $\sigma(W \cdot x + b) > 0.5$ it is a CAT!

Neural Network model



Neural Network forward propagation

- $z = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \\ w_{30} & w_{31} \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} + \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$ FIRST LINEAR LAYER
- $a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \tanh \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}$ TANH ACTIVATION FUNCTION
- $\mathrm{v} = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} + c$ SECOND LINEAR LAYER
- $\hat{y} = \sigma(v)$ SIGMOID ACTIVATION FUNCTION

PARAMETERS TO LEARN

Trainable parameters

- $W = \begin{pmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ w_{20} & w_{21} \\ w_{30} & w_{31} \end{pmatrix}$ Weight matrix first linear layer
- $b = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$ Bias vector first linear layer
- $\mathrm{T} = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 \end{pmatrix}$ Weight matrix second linear layer
- c Bias vector second linear layer

Neural Network: cost and loss

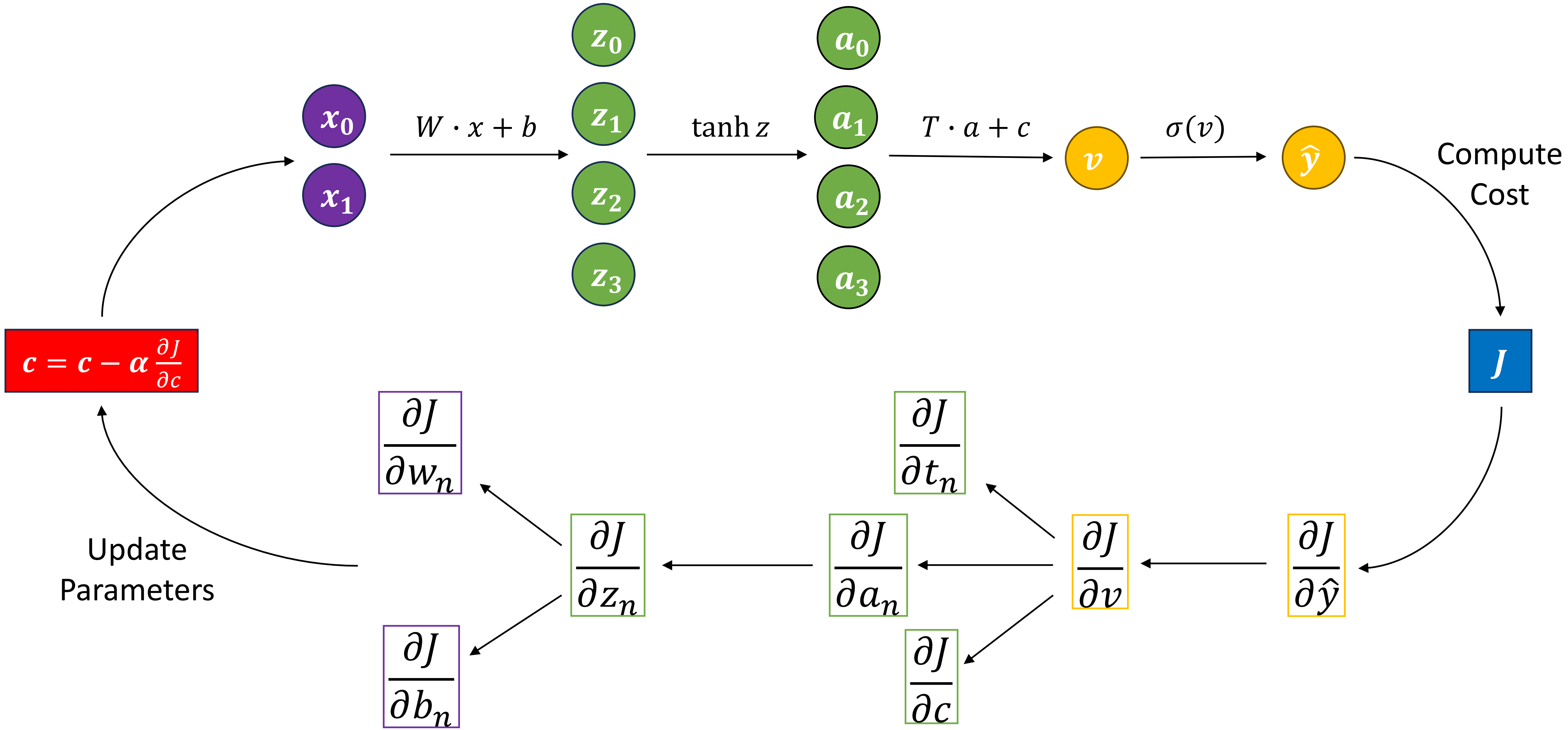
- COST and LOSS same as Logistic Regression:

$$J(W, b, T, c) = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

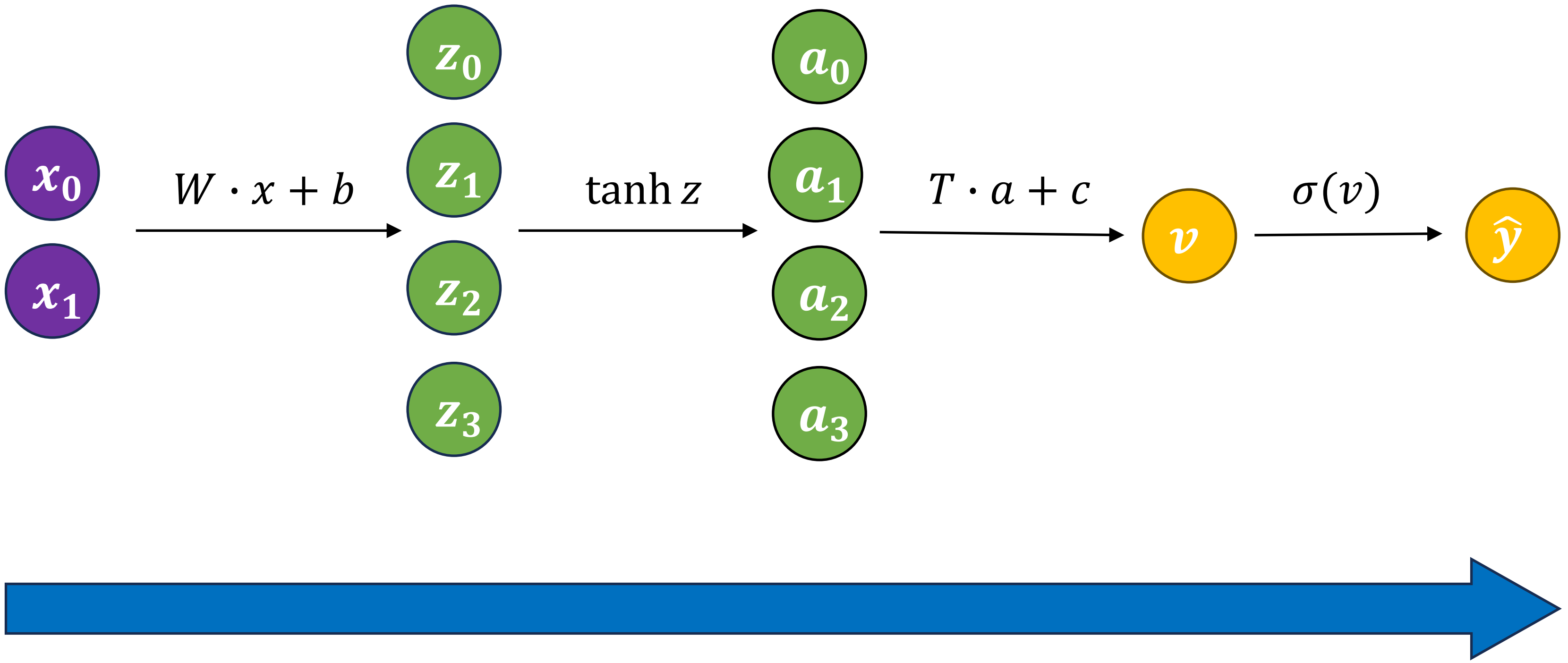
$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \cdot \log \hat{y}^{(i)} + (1 - y^{(i)}) \cdot \log\left(1 - \hat{y}^{(i)}\right))$$

- Gradient descent to find parameters W, b, T, c that minimize J.

Training Loop



Training Loop



FORWARD PROPAGATION

Training Loop