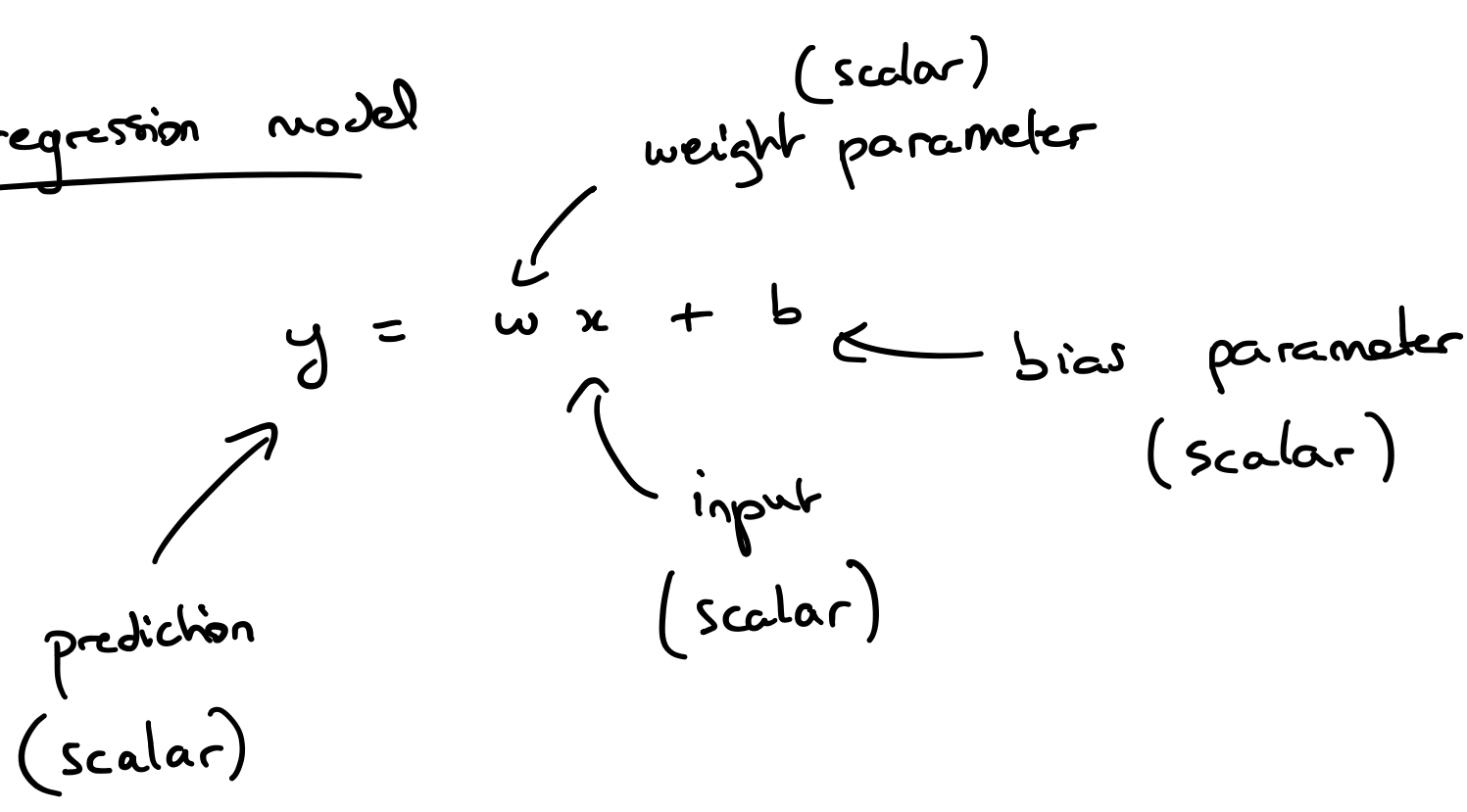


Linear regression model



Vector input:  $\vec{x} = (x_1, \dots, x_D)$

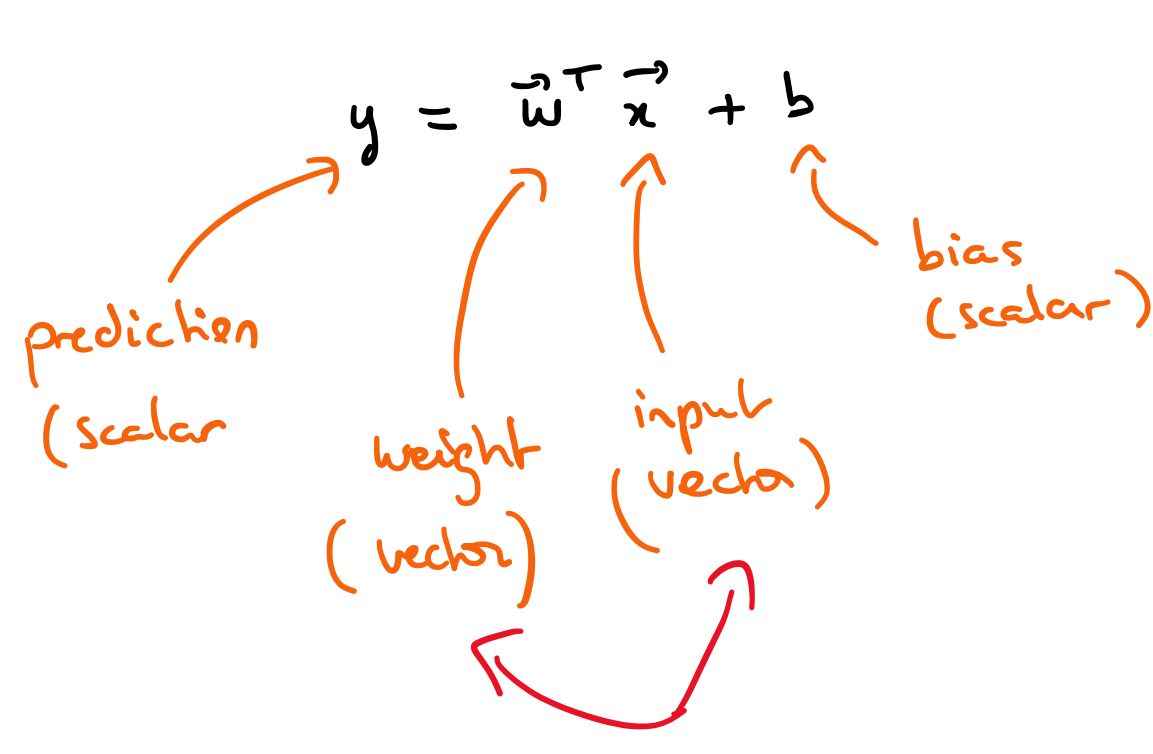
feature of the input

Python

```

y = b
for j in range(D):
    y += w[j] * x[j]
    
```

$$\vec{x} = (x_1, \dots, x_D) \quad \vec{w} = (w_1, \dots, w_D)$$



$$y = \vec{w}^T \vec{x} + b = (w_1, \dots, w_D) \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix} + b$$

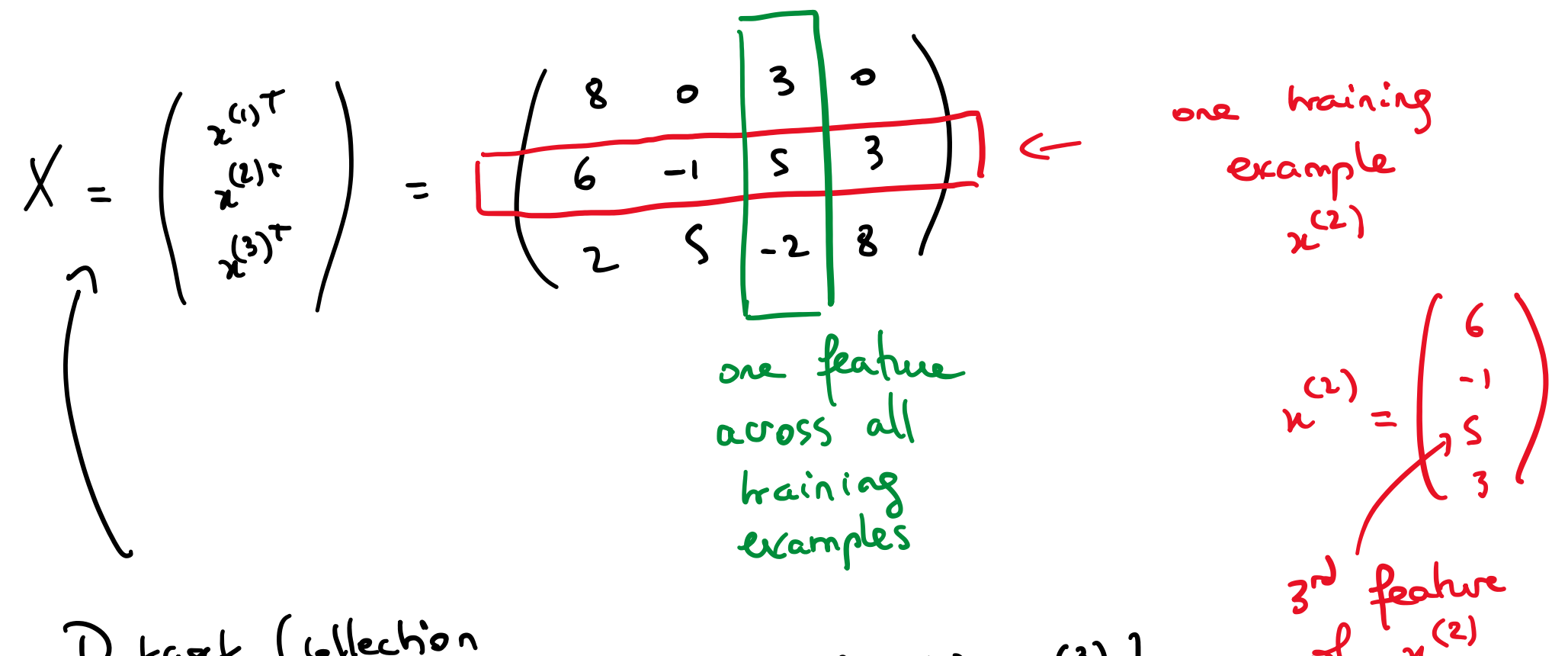
$$y = \sum_j w_j x_j + b$$

Python + NumPy:

$$y = \text{np.dot}(w, x) + b$$

np

Vectorize according to the  $n$  training examples:



Dataset (collection of training examples)

$$D = \{x^{(1)}, x^{(2)}, x^{(3)}\}$$

dataset matrix

$$X \vec{w} + b \mathbf{1} = \begin{pmatrix} w^T x^{(1)} + b \\ \vdots \\ w^T x^{(N)} + b \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{pmatrix} = \vec{y}$$

weight vector  $\vec{w}$   
 scalar bias  $b$   
 $\mathbf{1}$ : identity matrix  
 $N$ : nb training examples  
 predictions of our linear regression on each input of the dataset

each training examples has  $D=4$  features

$$X = \begin{pmatrix} x^{(1)T} \\ x^{(2)T} \\ x^{(3)T} \end{pmatrix} \quad \vec{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

$$X \vec{w} + b \mathbf{1} = \begin{pmatrix} x^{(1)T} \\ x^{(2)T} \\ x^{(3)T} \end{pmatrix} + \begin{pmatrix} b \\ \vdots \\ b \end{pmatrix}$$

$$= \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} + \begin{pmatrix} b \\ b \\ b \\ b \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{j=1}^4 w_j x_j^{(1)} + b \\ \sum_{j=1}^4 w_j x_j^{(2)} + b \\ \sum_{j=1}^4 w_j x_j^{(3)} + b \end{pmatrix}$$

$$= \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \end{pmatrix}$$

$y^{(2)}$  is prediction of linear model on  $x^{(2)}$

$$\vec{y} = X \vec{w} + b \mathbf{1}$$

$$\mathcal{E} = \frac{1}{2N} \|\vec{y} - \vec{t}\|^2$$

$$D = \{(x^{(i)}, t^{(i)})\}$$

predictions on entire dataset  
 labels for each training example

Python + numpy:

```

y = np.dot(X, w) + b
cost = np.sum((y - t)**2) / (2 * N)
    
```

cost = loss = error of the model