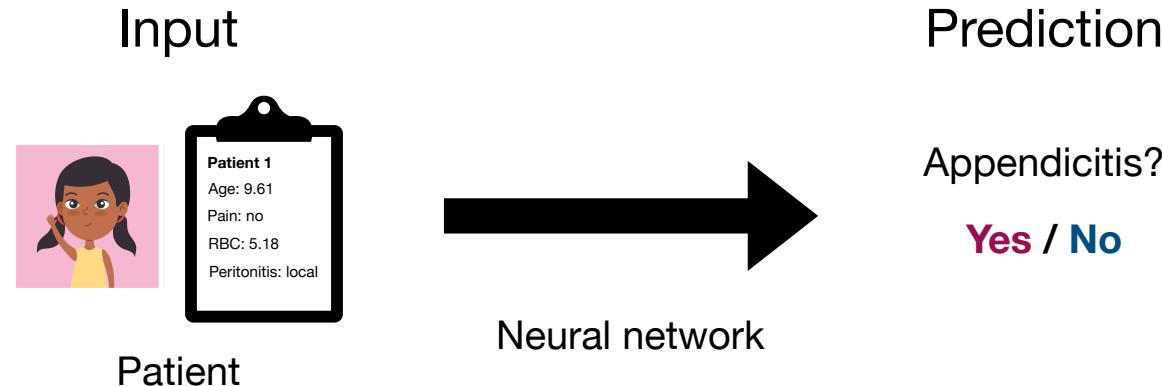


Linear regression, gradient descent & maximum likelihood

Setup

Example: Appendicitis diagnosis



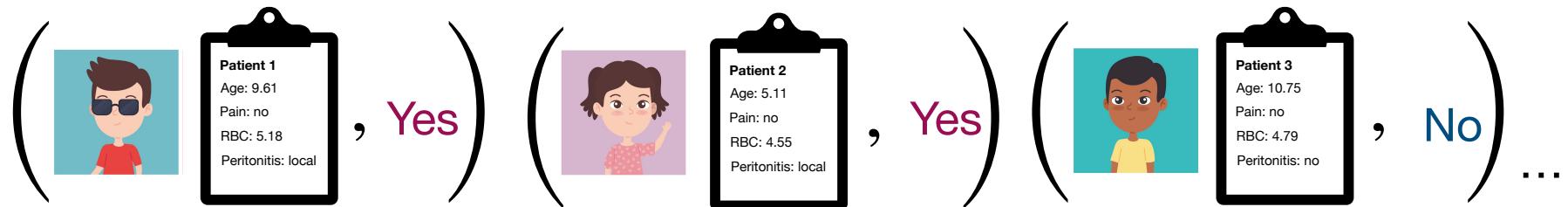
Some notation:

$$\text{Input: } \mathbf{x} \xrightarrow{\text{\textit{predict}}} \text{Output: } y, \quad y = f(\mathbf{x})$$

Prediction function

Dataset

Set of known inputs and **outputs**

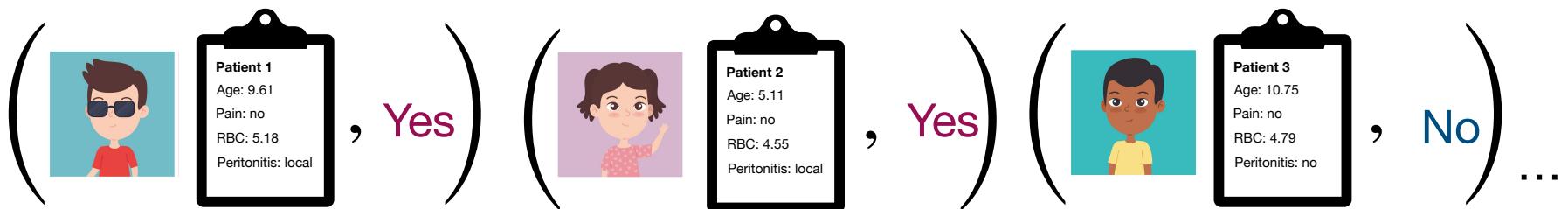


Some notation:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_N, y_N)\}$$

Dataset

Set of known inputs and **outputs**



Input table

Patients with abdominal pain					
	Age	Appendix Size	Migratory Pain	RBC Count	RBC Urine
0	9.61	9.0	no	5.18	high
1	5.11	7.0	no	4.55	medium
2	10.75	5.0	no	4.79	none
3	10.51	9.0	no	5.03	none
4	7.3	6.2	yes	4.64	low
5	15.21	8.5	yes	4.62	low
6	15.83	12.0	yes	4.33	high
7	9.58	7.0	yes	5.04	low
8	10.37	5.5	no	4.8	none
9	16.66	9.0	yes	5.31	none
10	14.52	4.5	yes	4.9	none
11	10.74	9.0	no	5.66	none
12	12.41	3.7	no	5.49	none
13	6.67	3.5	no	5.27	none
14	14.36	9.0	yes	4.84	low
15	9.04	5.3	yes	4.92	low
16	12.43	12.0	yes	4.62	none

Output table

	Diagnosis
0	appendicitis
1	no appendicitis
2	no appendicitis
3	no appendicitis
4	appendicitis
5	no appendicitis
6	no appendicitis
7	no appendicitis
8	no appendicitis
9	appendicitis
10	appendicitis
11	no appendicitis
12	no appendicitis
13	no appendicitis
14	appendicitis
15	no appendicitis
16	appendicitis

Mathematical abstraction

As table

	Patients with abdominal pain							
	Age	Appendix Size	Height	Weight	RBC Count	Temperature	WBC Count	
0	16.66		9.0	174.0	65.0	5.31	36.6	6.6
1	10.74		9.0	146.0	57.5	5.66	37.3	10.2
2	9.04		5.3	134.0	29.4	4.92	36.0	5.1
3	11.3		5.0	155.0	54.5	4.79	37.7	10.3
4	11.3		6.2	123.0	23.5	4.64	37.4	21.1
5	5.11		7.0	116.0	22.0	4.55	40.2	19.4
6	14.36		9.0	163.0	50.0	4.84	37.5	14.3
7	9.61		9.0	140.0	29.2	5.18	38.7	14.3
8	15.83		12.0	153.0	59.0	4.33	36.7	12.8
9	9.58		7.0	132.0	24.7	5.04	38.4	13.5
10	10.37		5.5	156.0	39.0	4.8	37.4	5.6
11	14.52		4.5	181.0	55.0	4.9	37.0	9.0
12	12.41		3.7	150.5	42.5	5.49	37.2	9.1
13	6.67		3.5	124.0	38.5	5.27	39.6	16.8
14	15.21		8.5	155.0	85.0	4.62	36.8	12.4
15	12.43		12.0	157.0	46.0	4.62	37.1	16.4
16	10.51		9.0	134.5	27.0	5.03	37.4	12.8

Feature

$N \times d$ Matrix: $\mathbf{X} \in \mathbb{R}^{N \times d}$

As matrix

Observation

$\mathbf{X} =$

16.66	9.0	174.0	65.0	5.31	36.6	6.6	N (Obs.)
10.74	9.0	146.0	57.5	5.66	37.3	10.2	
9.04	5.3	134.0	29.4	4.92	36.0	5.1	
10.75	5.0	155.0	54.5	4.79	37.7	10.3	
7.3	6.2	123.0	23.5	4.64	37.4	21.1	
5.11	7.0	116.0	22.0	4.55	40.2	19.4	
14.36	9.0	163.0	50.0	4.84	37.5	14.3	
9.61	9.0	140.0	29.2	5.18	38.7	14.3	
15.83	12.0	153.0	59.0	4.33	36.7	12.8	
9.58	7.0	132.0	24.7	5.04	38.4	13.5	
10.37	5.5	156.0	39.0	4.8	37.4	5.6	
14.52	4.5	181.0	55.0	4.9	37.0	9.0	
12.41	3.7	150.5	42.5	5.49	37.2	9.1	
6.67	3.5	124.0	38.5	5.27	39.6	16.8	
15.21	8.5	155.0	85.0	4.62	36.8	12.4	
12.43	12.0	157.0	46.0	4.62	37.1	16.4	
10.51	9.0	134.5	27.0	5.03	37.4	12.8	

Feature

d
(features)

Mathematical abstraction

Labels become a vector

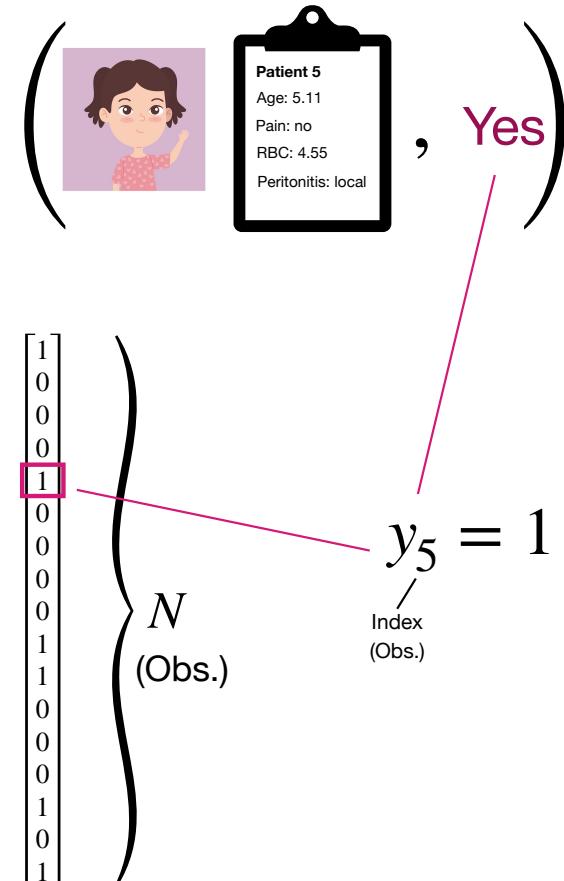
Diagnosis	
	Diagnosis
0	appendicitis
1	no appendicitis
2	no appendicitis
3	no appendicitis
4	appendicitis
5	no appendicitis
6	no appendicitis
7	no appendicitis
8	no appendicitis
9	appendicitis
10	appendicitis
11	no appendicitis
12	no appendicitis
13	no appendicitis
14	appendicitis
15	no appendicitis
16	appendicitis

Output table

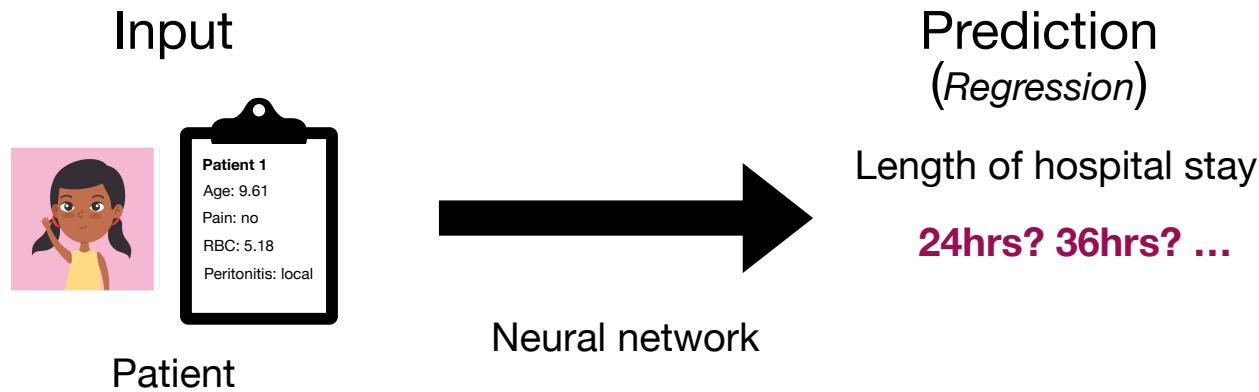
Converted

Diagnosis	
	Diagnosis
0	1
1	0
2	0
3	0
4	1
5	0
6	0
7	0
8	0
9	1
10	0
11	0
12	0
13	0
14	1
15	0
16	1

$y =$



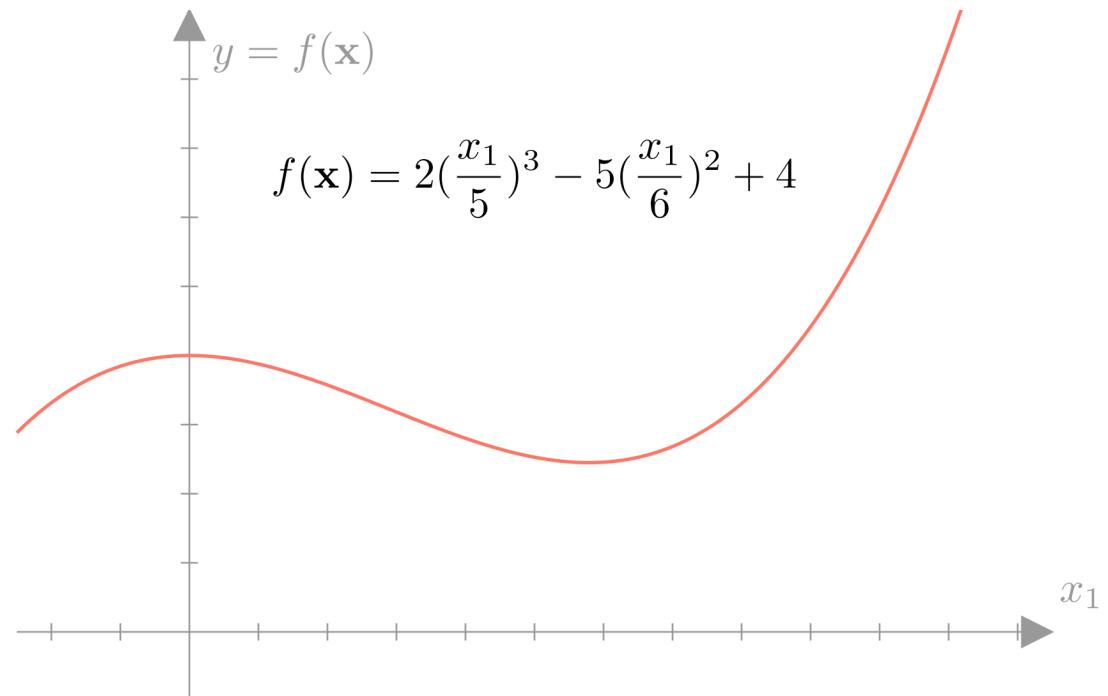
Quantitative outputs: *Regression*



$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$$

Output domain: $y \in \mathbb{R}$

Input: $\mathbf{x} \xrightarrow{predict}$ Output: y , $y = f(\mathbf{x})$
Prediction function



Honda Accord:

Weight:	2500 lbs
Horsepower:	123 HP
Displacement:	2.4 L
0-60mph:	7.8 Sec

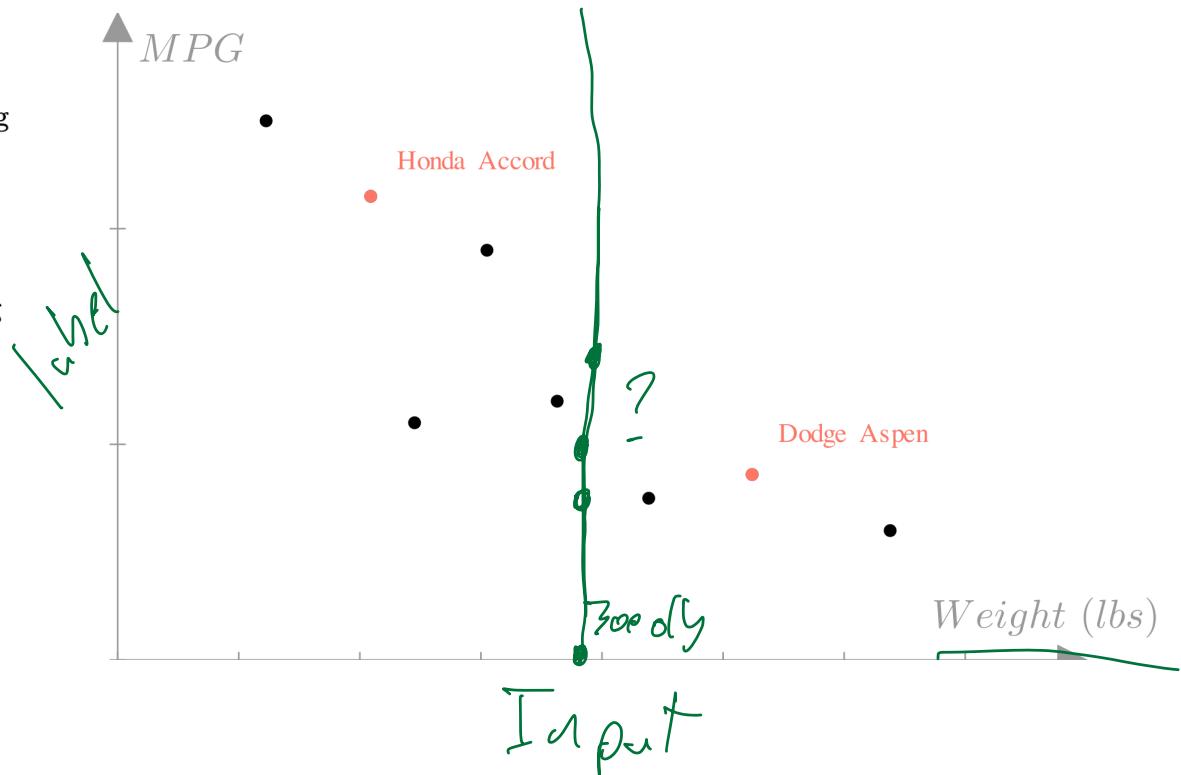
→ MPG: 33mpg

Dodge Aspen:

Weight:	3800 lbs
Horsepower:	155 HP
Displacement:	3.2 L
0-60mph:	6.8 Sec

→ MPG: 21mpg

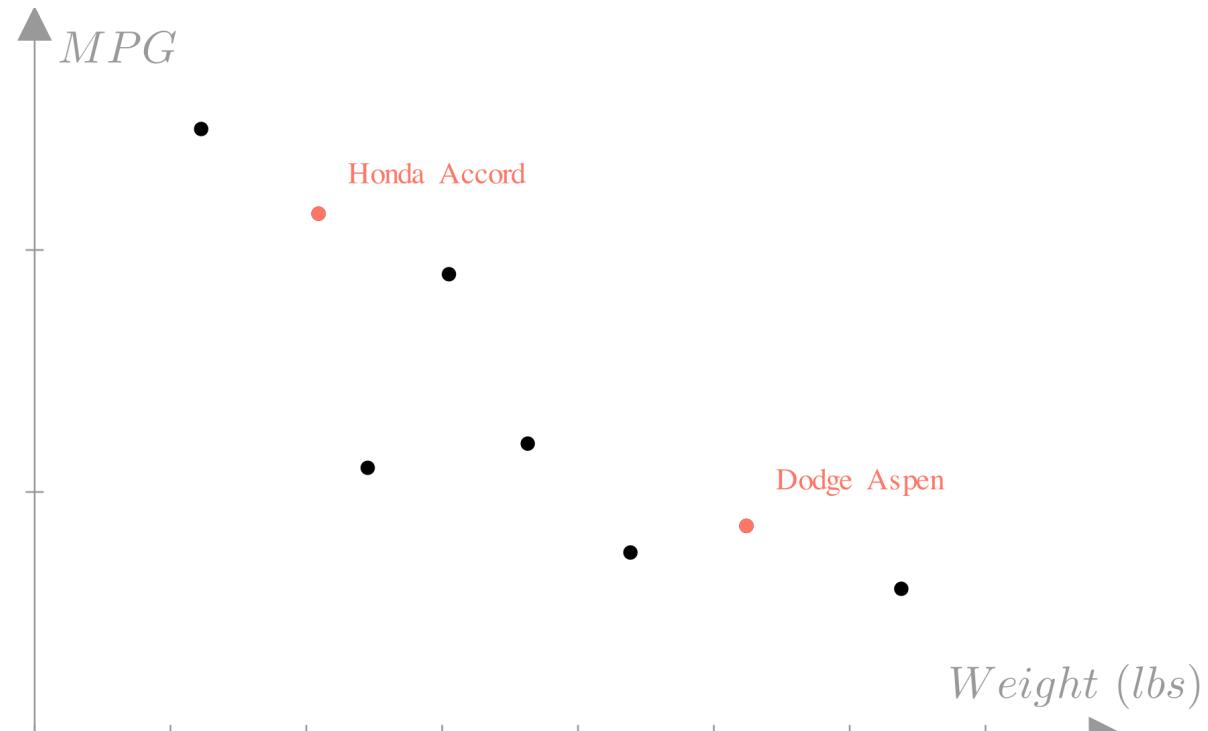
⋮ ⋮

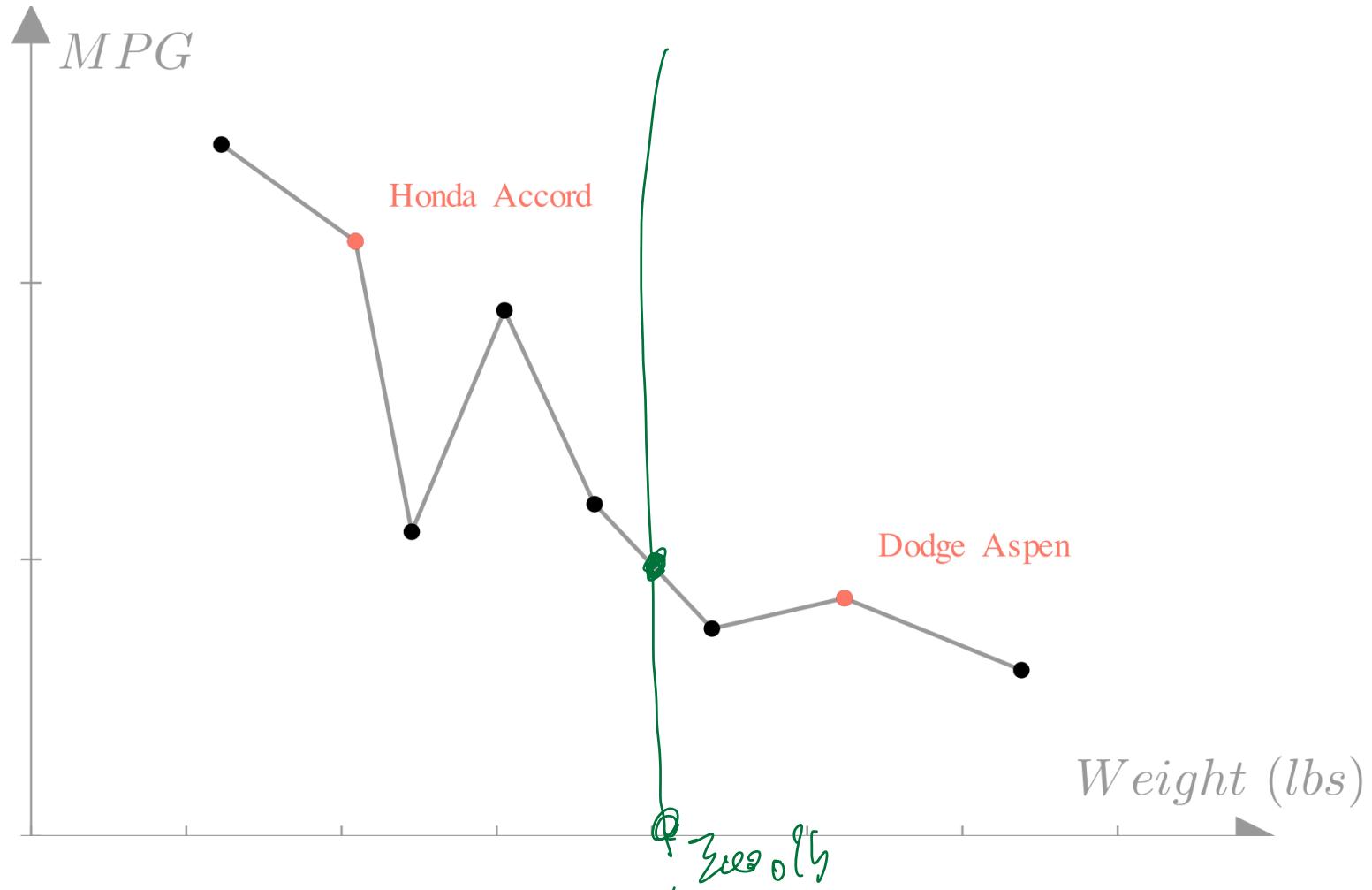


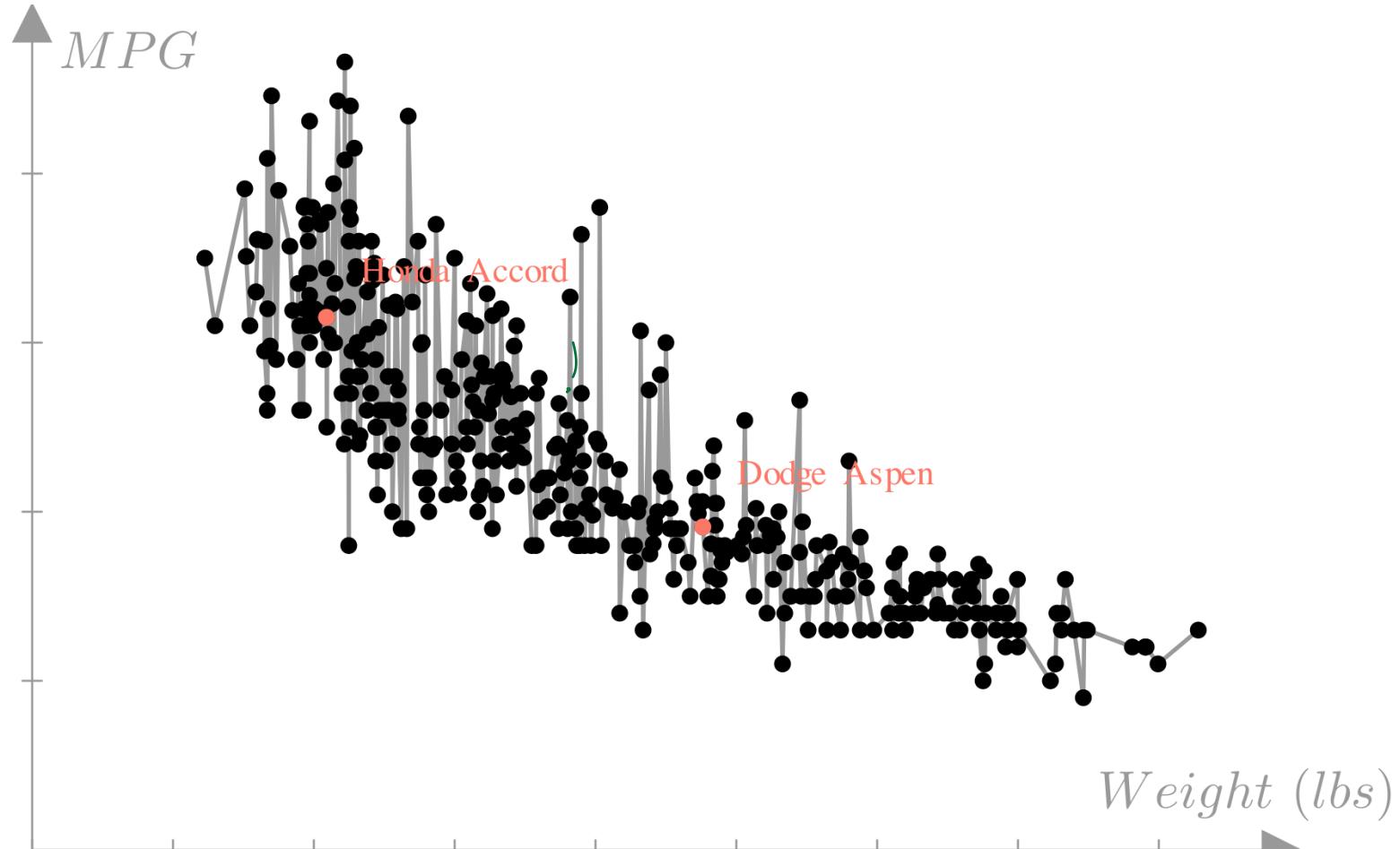
Dataset: $\mathcal{D} = \{(\mathbf{x}_i, y_i) \text{ for } i \in 1 \dots N\}$

Input: $\mathbf{x}_i = \begin{bmatrix} \text{Weight} \\ \text{Horsepower} \\ \text{Displacement} \\ \text{0-60mph} \end{bmatrix}, \text{ Output: } y_i = MPG$

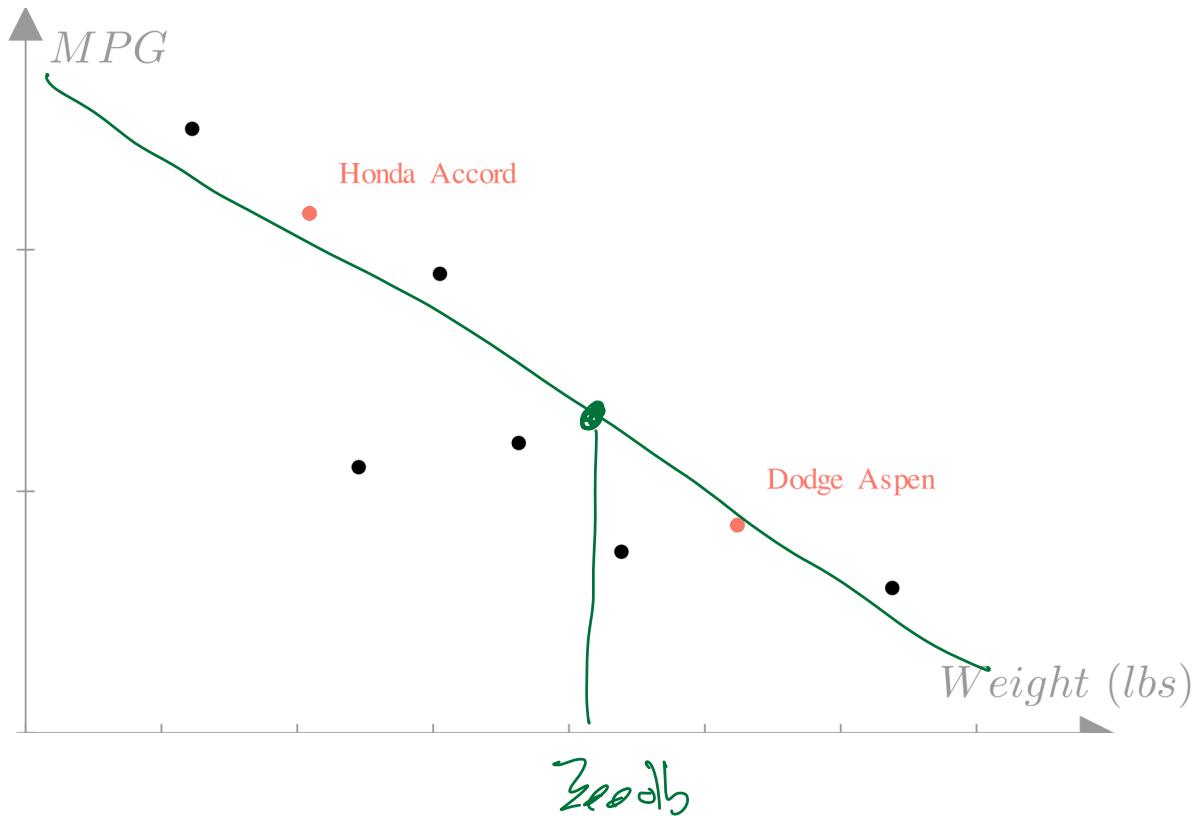
$$mpg = f(\text{weight, horsepower...})$$







Linear regression



A *linear function* is any function f where the following conditions always hold:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

and

$$f(a\mathbf{x}) = a f(\mathbf{x})$$

For a linear function, the output can be defined as a weighted sum of the inputs. In other words a linear function of a vector can always be written as:

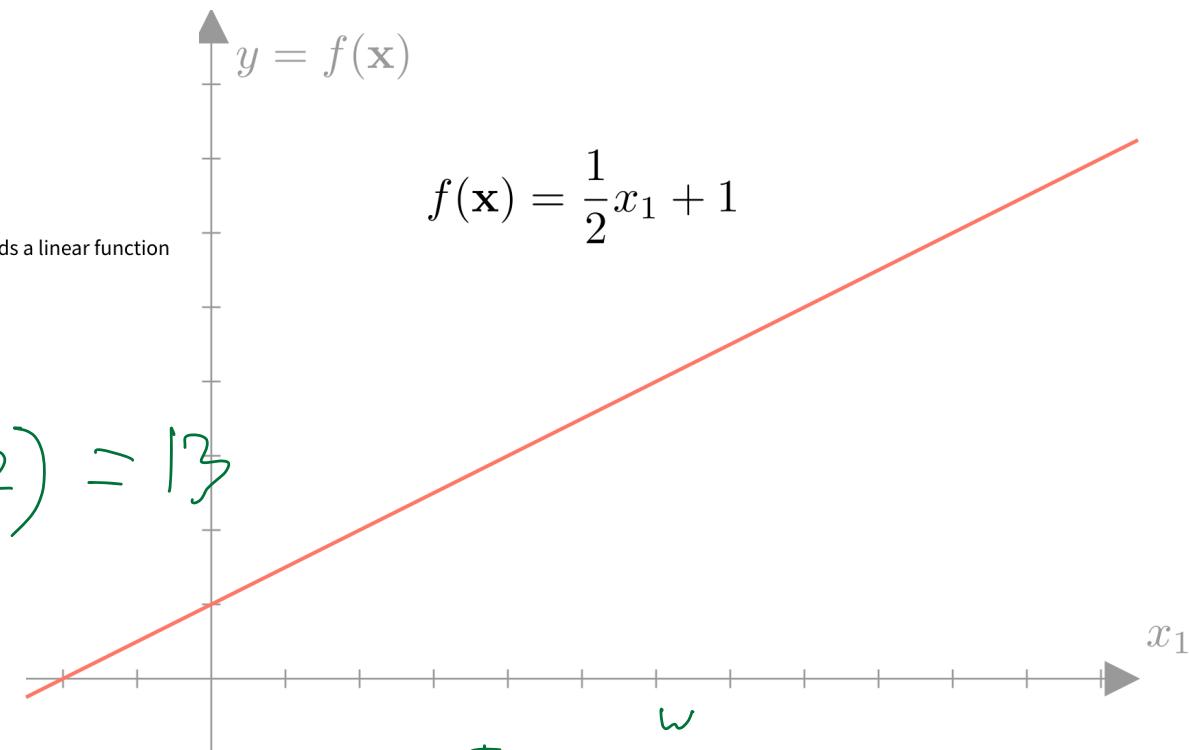
$$f(\mathbf{x}) = \sum_{i=1}^n x_i w_i$$

$$f\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = 3(1) + 5(2) = 13$$

$\uparrow \quad \uparrow$
 $w_1 \quad w_2$

$$\mathbf{w} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$$

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} \hat{=} \mathbf{x}^\top \mathbf{w}$$



$$\begin{bmatrix} \rightarrow \\ \mathbf{x}^\top \end{bmatrix} \begin{bmatrix} \downarrow \\ \mathbf{w} \end{bmatrix}$$

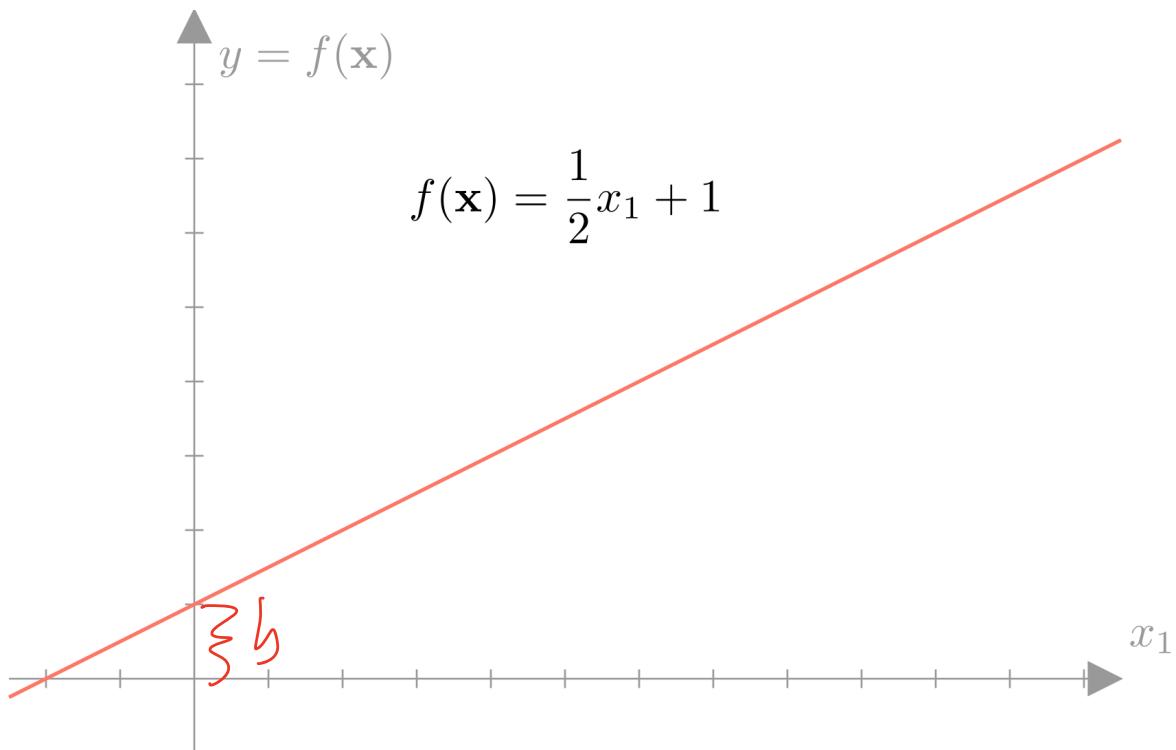
We can add an offset b to create an *affine* function:

$$f(\mathbf{x}) = \sum_{i=1}^n x_i w_i + b$$

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w} + b$$

In numpy we can easily write a function of this form:

```
def f(x):
    w = np.array([-0.6, -0.2])
    b = -1
    return np.dot(x, w) + b
```



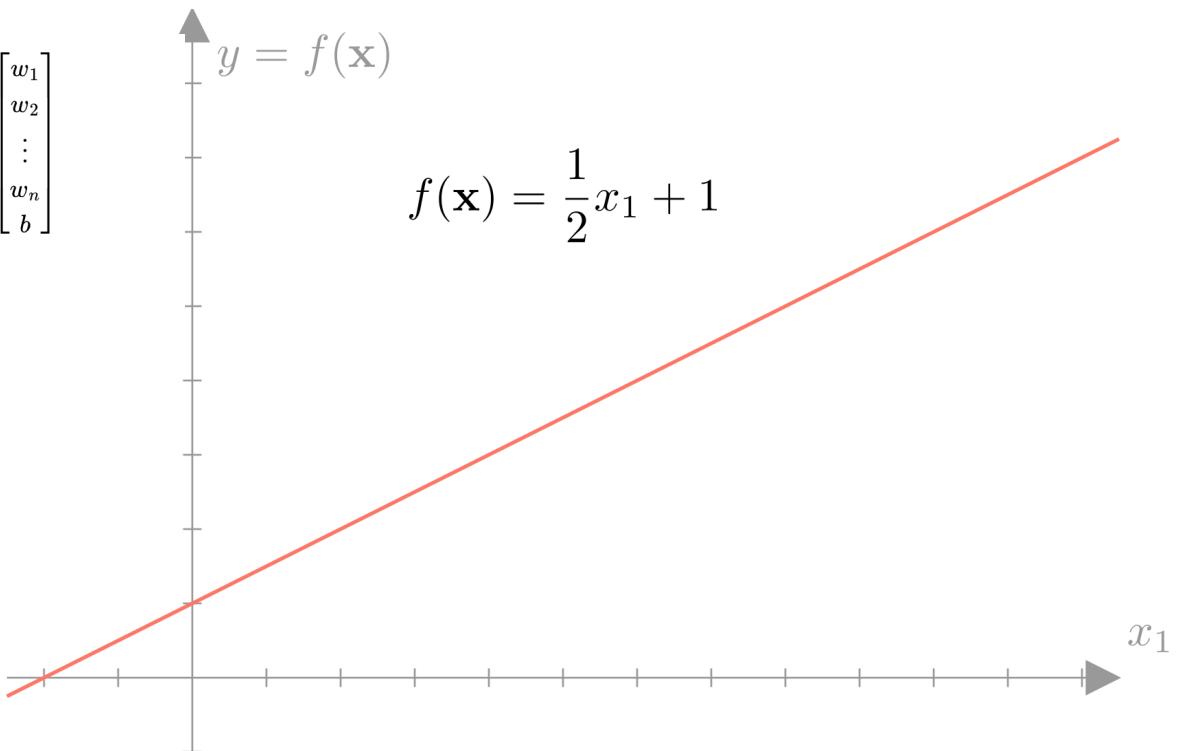
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \longrightarrow \mathbf{x}_{aug} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \longrightarrow \mathbf{w}_{aug} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ b \end{bmatrix}$$

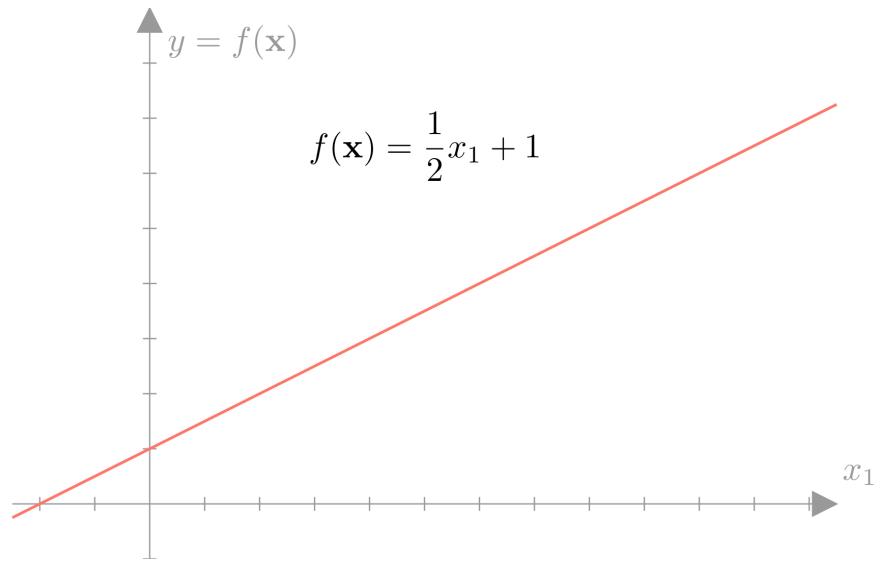
$$y = f(\mathbf{x})$$

$$f(\mathbf{x}) = \frac{1}{2}x_1 + 1$$

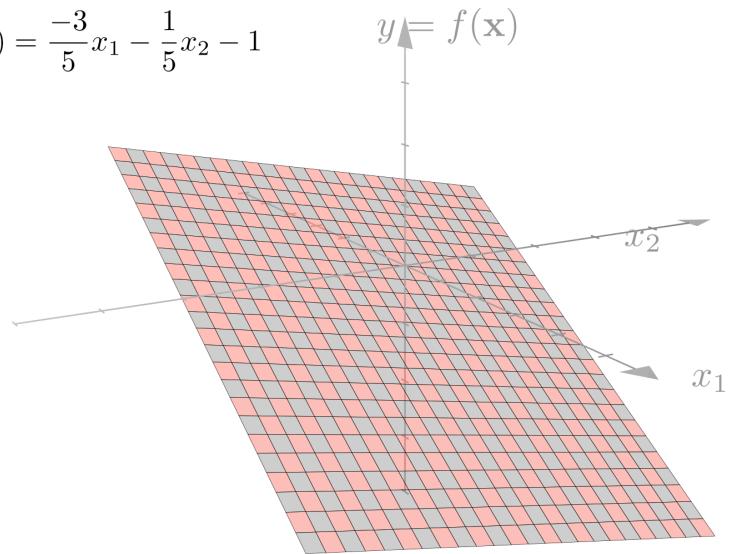
We can easily see then that using this notation:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b = \mathbf{x}_{aug}^T \mathbf{w}_{aug}$$





$$f(\mathbf{x}) = \frac{-3}{5}x_1 - \frac{1}{5}x_2 - 1$$



Linear regression is the approach of modeling an unknown function with a linear function. From our discussion of linear functions, we know that this means that we will make predictions using a function of the form:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} = \sum_{i=1}^n x_i w_i$$

Meaning that the output will be a weighted sum of the *features* of the input. In the case of our car example, we will make predictions as:

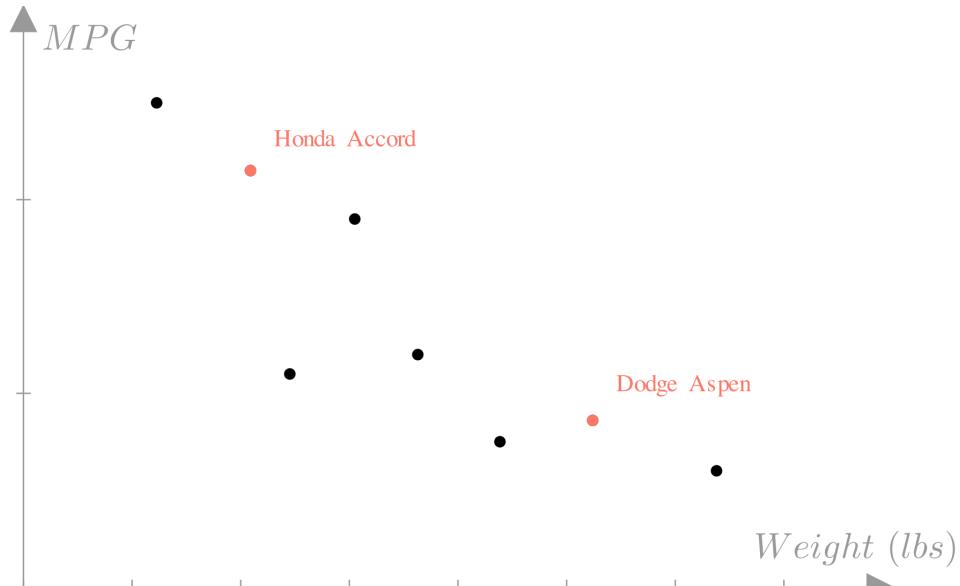
$$\text{Predicted MPG} = f(\mathbf{x}) =$$

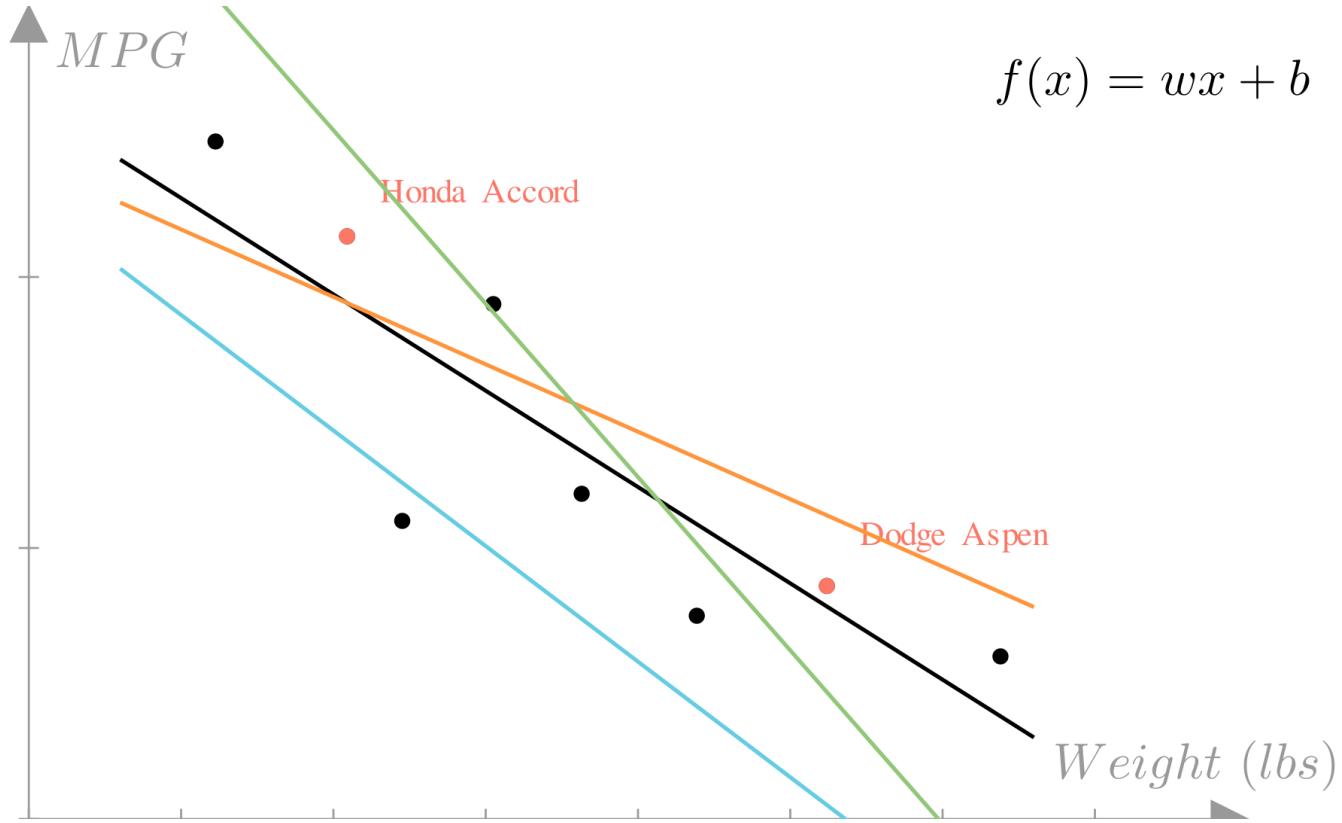
$$(\text{weight})w_1 + (\text{horsepower})w_2 + (\text{displacement})w_3 + (0\text{-}60\text{mph})w_4 + b$$

Or in matrix notation:

$$f(\mathbf{x}) = \begin{bmatrix} \text{Weight} \\ \text{Horsepower} \\ \text{Displacement} \\ 0\text{-}60\text{mph} \\ 1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ b \end{bmatrix}$$

Observation \nwarrow Parameters (w, b)





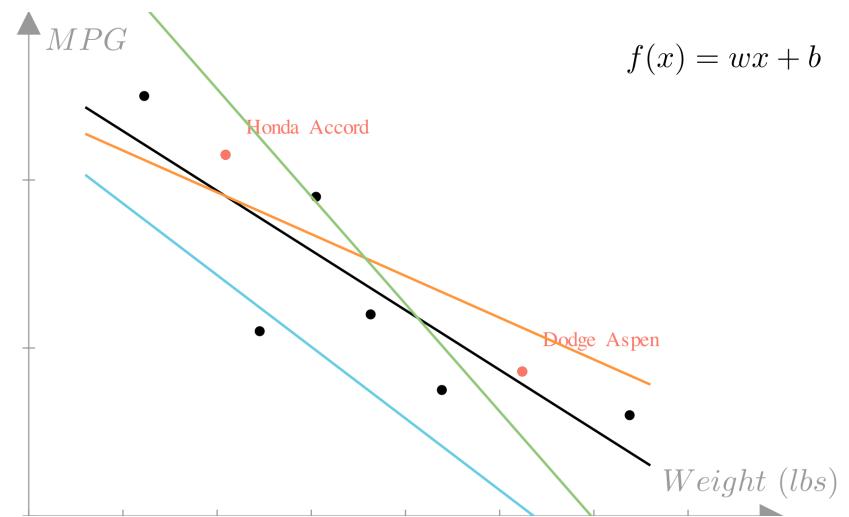
$$f(\mathbf{x}) = \sum_{i=1}^n x_i w_i + b$$

We typically refer to \mathbf{w} specifically as the **weight vector** (or weights) and b as the **bias**. To summarize:

Affine function: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b$, **Parameters:** (Weights: \mathbf{w} , Bias: b)

```
class Regression:
    def __init__(self, weights):
        self.weights = weights

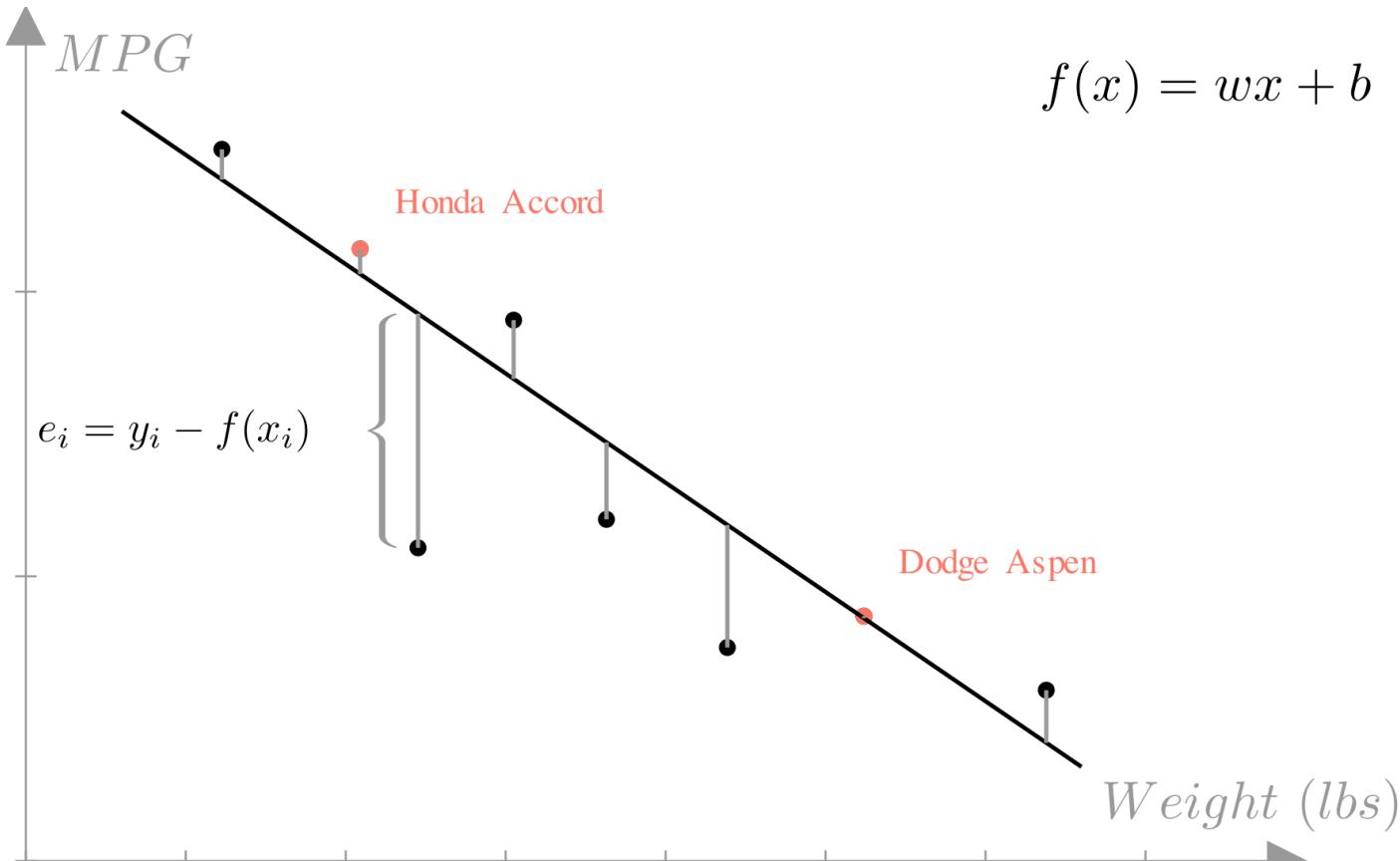
    def predict(self, x):
        return np.dot(x, self.weights)
```



The **residual** or **error** of a prediction is the difference between the prediction and the true output:

$$e_i = y_i - f(\mathbf{x}_i)$$

$$f(x) = wx + b$$



$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_N, y_N)\}$$

mean squared error
prediction

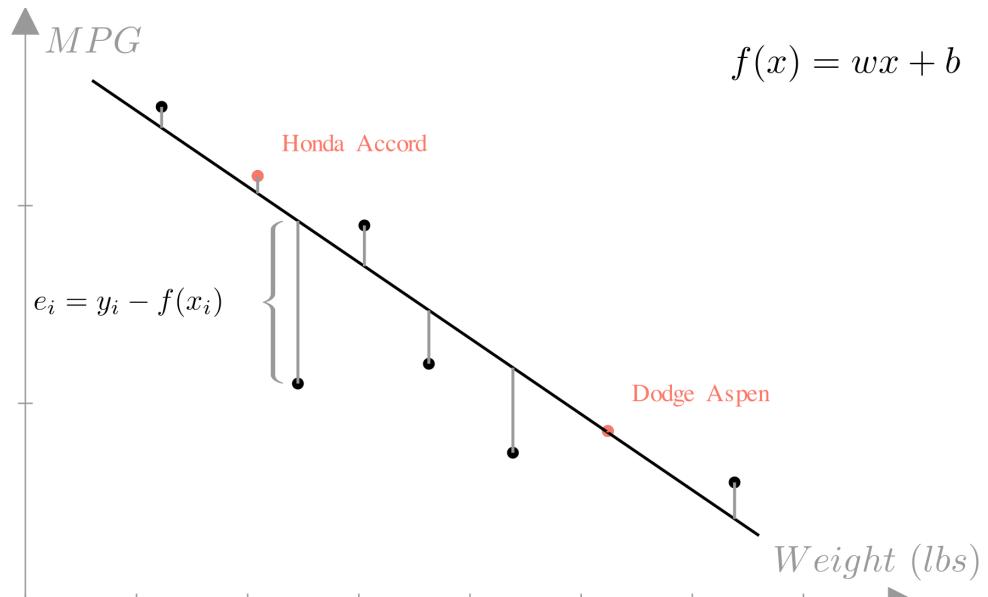
$$MSE = \frac{1}{N} \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2 = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

w
mean
label

$$\frac{1}{N} \sum_{i=1}^N |f(\mathbf{x}_i) - y_i|$$

Mean absolute error

Loss function



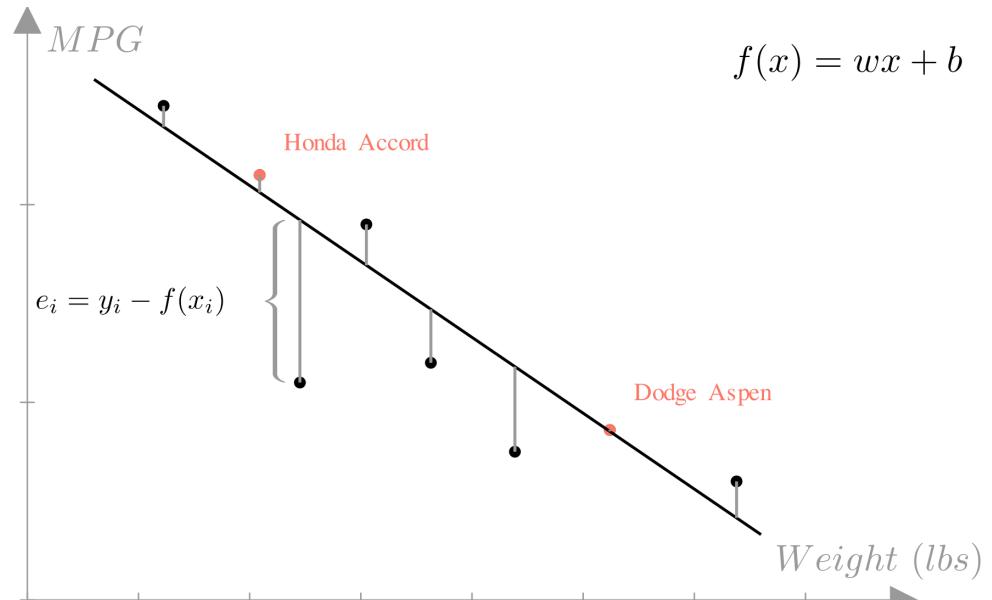
$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_N, y_N)\}$$

$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix}$

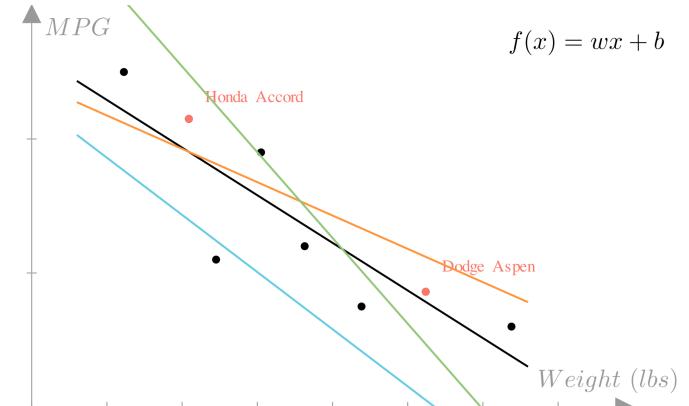
feature
weight
Hd
y = $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$ mpg
obs
aspen

$$MSE(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

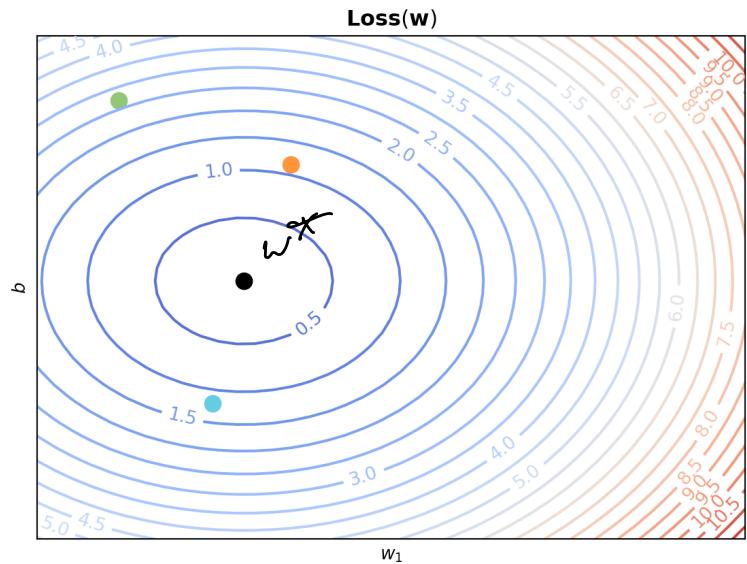
$f(x) = \mathbf{x}^T \mathbf{w}$



$$\text{Loss}(\mathbf{w}) = MSE(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$



$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \text{Loss}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$



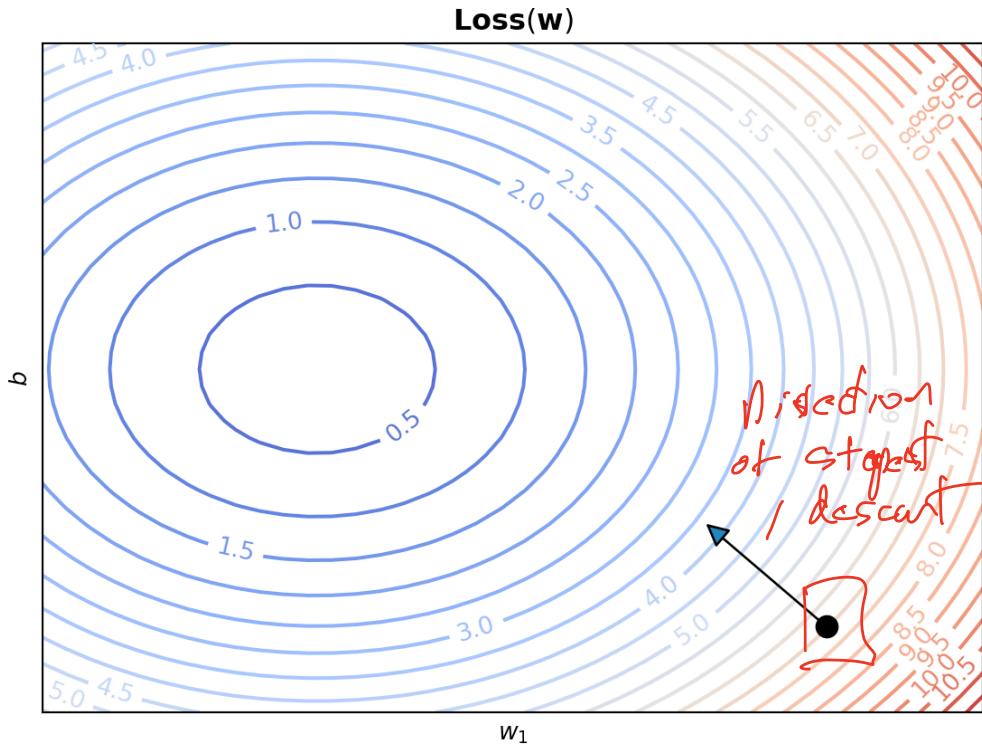
$$\text{Find: } \mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} f(\mathbf{w})$$

Initial guess

)

$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} + \mathbf{g}$$

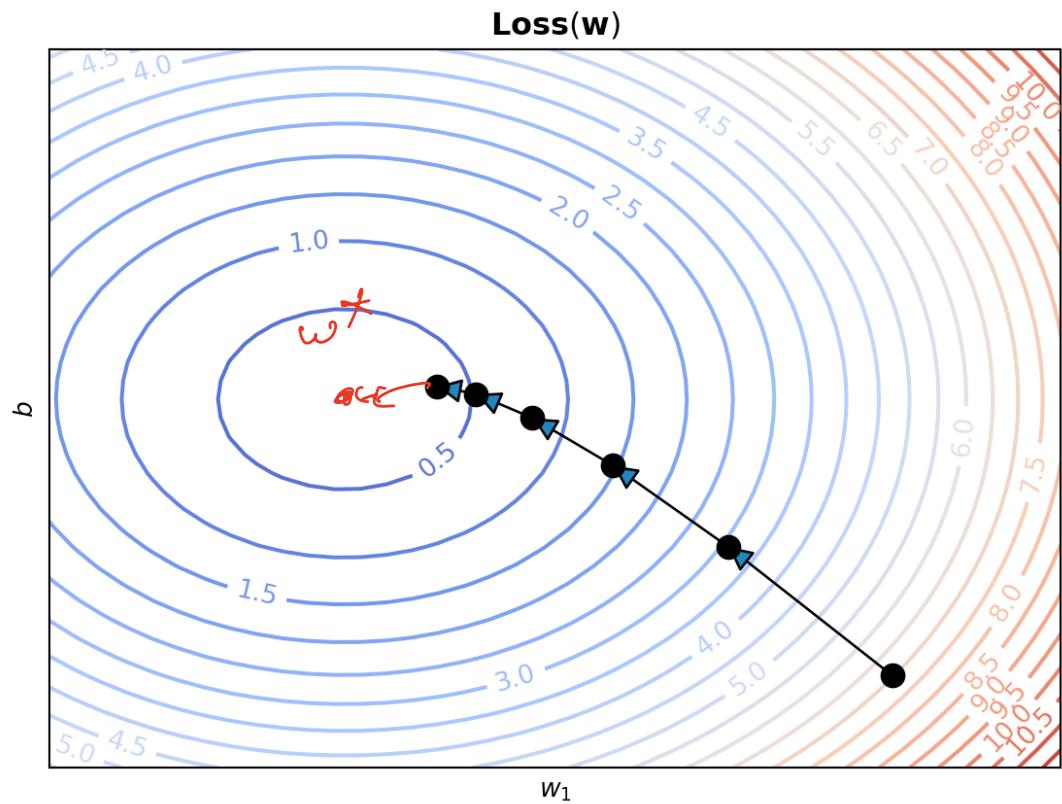
$$\text{Loss}(w^{(1)}) < \text{loss}(w^{(0)})$$



Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

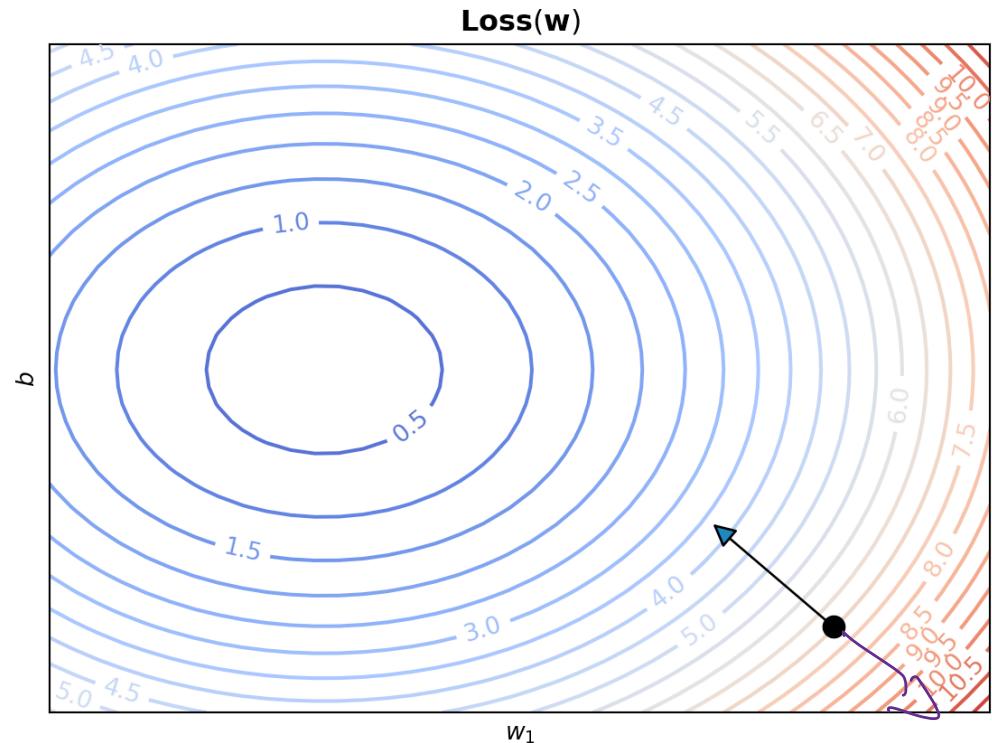
$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} + \mathbf{g}$$

Gradient descent



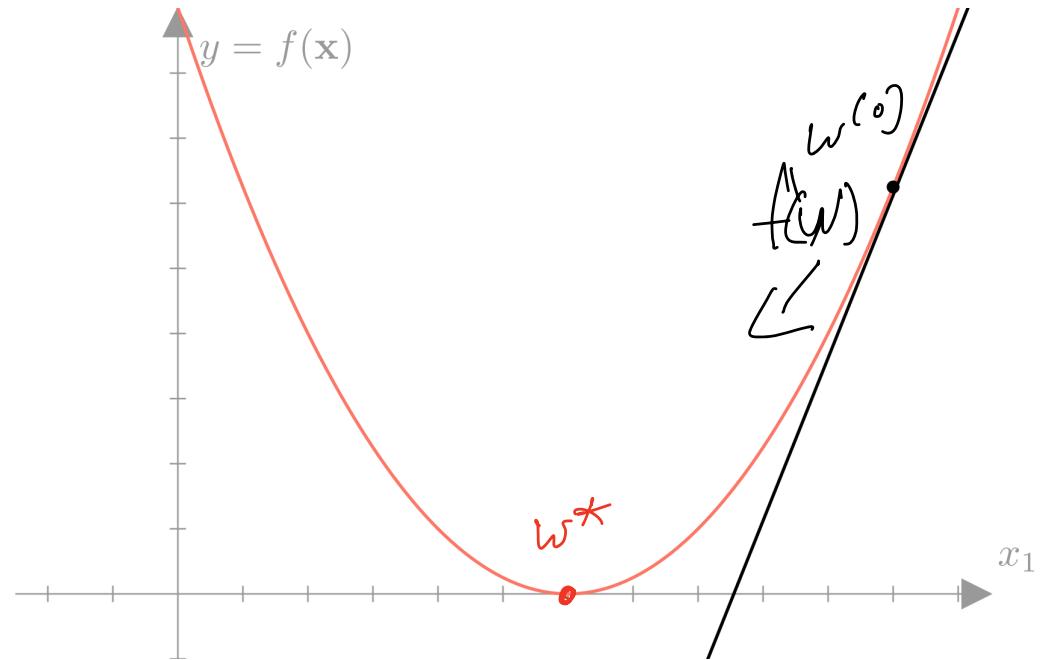
Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} - \nabla f(\mathbf{w}^{(0)})$$



Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

$$\mathbf{w}^{(1)} \leftarrow \mathbf{w}^{(0)} + \mathbf{g}$$



The **gradient** of a vector-input function is a vector such that each element is the partial derivative of the function with respect to the corresponding element of the input vector. We'll use the same notation as derivatives for gradients.

$$f(\mathbf{x}) \quad \tilde{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ i \end{bmatrix} \quad \frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix} \quad f(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \frac{\partial}{\partial x_1} \mathbf{x}^T \mathbf{x} = \underbrace{\mathbf{x} \cdot \mathbf{x}}_{\sum_{i=1}^3 x_i x_i} = \sum_{i=1}^3 x_i^2 = x_1^2 + x_2^2 + x_3^2$$

$\frac{d}{dx_1} = 2x_1$

The **gradient** of a vector-input function is a vector such that each element is the partial derivative of the function with respect to the corresponding element of the input vector. We'll use the same notation as derivatives for gradients.

$$\frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \frac{\partial}{\partial x_1} \mathbf{x}^T \mathbf{x} = 2x_1, \quad \frac{\partial}{\partial x_i} = 2x_i$$

gradient

$$\frac{d}{d\mathbf{x}} \mathbf{x}^T \mathbf{x} = \left[\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3} \right] = \begin{bmatrix} 2x_1 \\ 2x_2 \\ 2x_3 \end{bmatrix} = 2 \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 2\vec{x}$$

The **gradient** of a vector-input function is a vector such that each element is the partial derivative of the function with respect to the corresponding element of the input vector. We'll use the same notation as derivatives for gradients.

$$\frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix}$$

\downarrow
 $2f_1$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \frac{\partial}{\partial x_1} f(\mathbf{x}^T \mathbf{x}) = f'(x_1^2 + x_2^2 + x_3^2) = f'(x_1^2 + x_2^2 + x_3^2) \left(\frac{\partial}{\partial x_1} x^T x \right)$$

$$= 2x_1 \cdot f'(x^T x)$$

$$\frac{d}{d\mathbf{x}} f(\mathbf{x}^T \mathbf{x}) = 2f'(x^T x) \underline{x}$$

& scalar

The **gradient** of a vector-input function is a vector such that each element is the partial derivative of the function with respect to the corresponding element of the input vector. We'll use the same notation as derivatives for gradients.

$$\frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix}$$

$$\frac{\partial}{\partial x} \sum f(x) = \sum \frac{\partial}{\partial x} f(x)$$

↙

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \frac{\partial}{\partial x_1} \sum_{i=1}^3 f(x_i) = \sum_{i=1}^3 \frac{\partial}{\partial x_i} f(x_i)$$

The **gradient** of a vector-input function is a vector such that each element is the partial derivative of the function with respect to the corresponding element of the input vector. We'll use the same notation as derivatives for gradients.

$$\frac{\partial f}{\partial x} = f'(x)$$

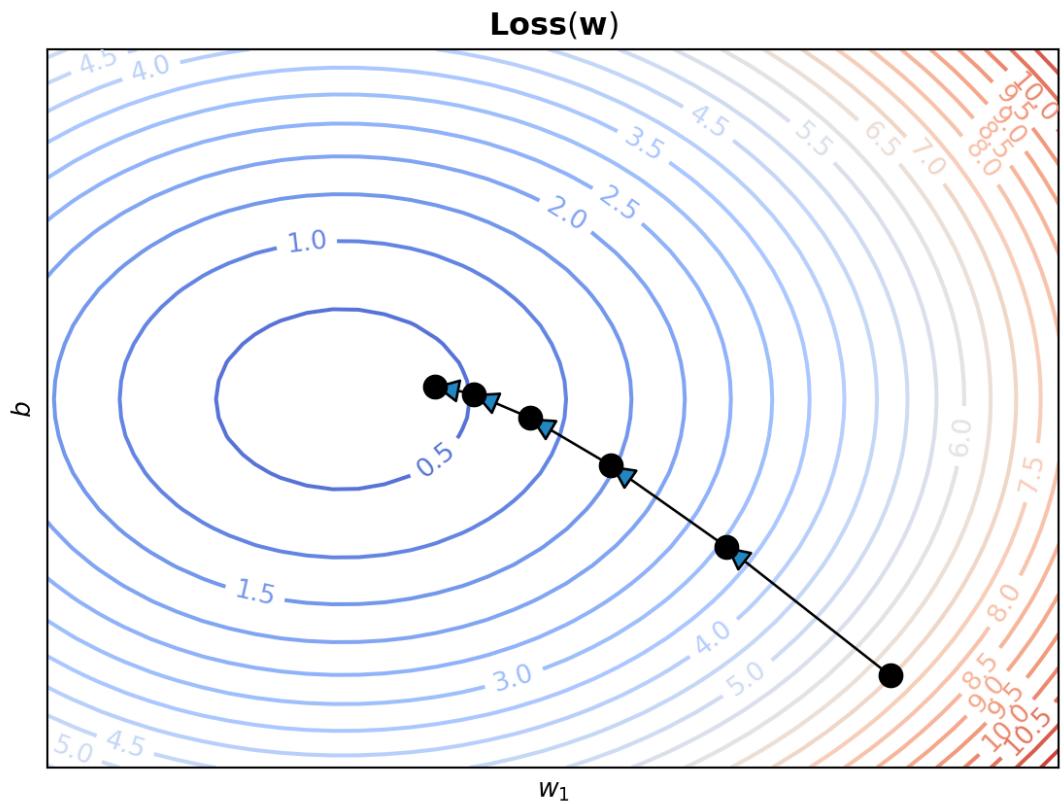
$$\frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix}$$

$$\frac{df}{d\mathbf{x}} = \nabla f(\mathbf{x})$$

$$\frac{df}{d\mathbf{x}} = \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}), \quad \frac{df}{d\mathbf{y}} = \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})$$

Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

For i in 1,...,T : $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \nabla f(\mathbf{w}^{(i)})$



$$\nabla_{\mathbf{w}} \text{MSE}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{d}{d\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2 \right) \stackrel{\text{pred. } f(x) = \mathbf{x}^T \mathbf{w}}{}$$

$$= \frac{1}{N} \left(\frac{d}{d\mathbf{w}} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2 \right) = \frac{1}{N} \sum_{i=1}^N \frac{d}{d\mathbf{w}} (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

$$= \frac{1}{N} \sum_{i=1}^N 2(\mathbf{x}_i^T \mathbf{w} - y_i) \frac{d}{d\mathbf{w}} (\mathbf{x}_i^T \mathbf{w} - y_i)$$

$$= \frac{1}{N} \sum_{i=1}^N 2(\mathbf{x}_i^T \mathbf{w} - y_i) \frac{d}{d\mathbf{w}} \mathbf{x}_i^T \mathbf{w}$$

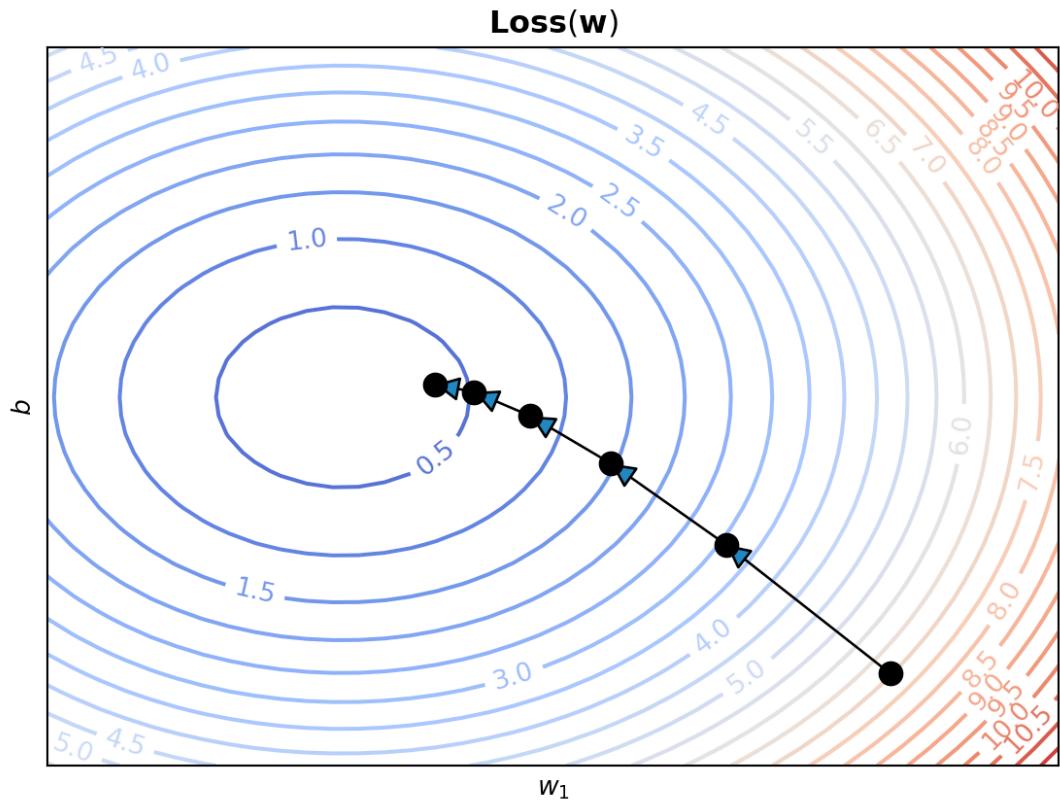
$$\frac{\partial}{\partial w_a} \sum_{j=1}^d x_{ij} w_j = \underbrace{\sum_{j=1}^d \frac{\partial}{\partial w_a} x_{ij} w_j}_{=0 \text{ if } a \neq j}$$

$$= \frac{1}{N} \sum_{i=1}^N 2(\mathbf{x}_i^T \mathbf{w} - y_i) \mathbf{x}_i^T$$

Find: $\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} f(\mathbf{w})$

For i in $1, \dots, T$: $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \nabla f(\mathbf{w}^{(i)})$

$$\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \frac{1}{N} \sum_{i=1}^N 2(x_i^\top \mathbf{w}^{(i)} - y_i) x_i$$



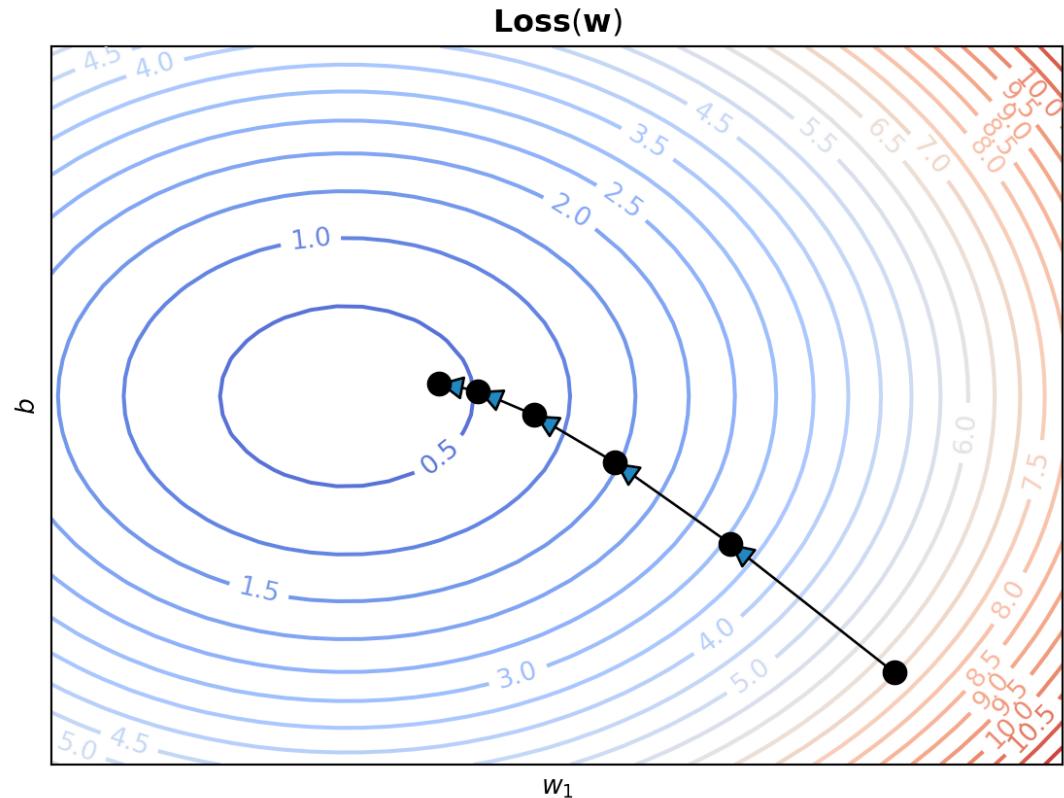
Recall that it's minimum value \mathbf{w}^* , a function f must have a gradient of $\mathbf{0}$.

$$\nabla f(\mathbf{w}^*) = \mathbf{0}$$

It follows that:

$$\mathbf{w}^* = \mathbf{w}^* - \nabla f(\mathbf{w}^*)$$

While $\nabla f(\mathbf{w}^{(i)}) \neq \mathbf{0}$: $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \nabla f(\mathbf{w}^{(i)})$



Recall that it's minimum value \mathbf{w}^* , a function f must have a gradient of $\mathbf{0}$.

$$\nabla f(\mathbf{w}^*) = \mathbf{0}$$

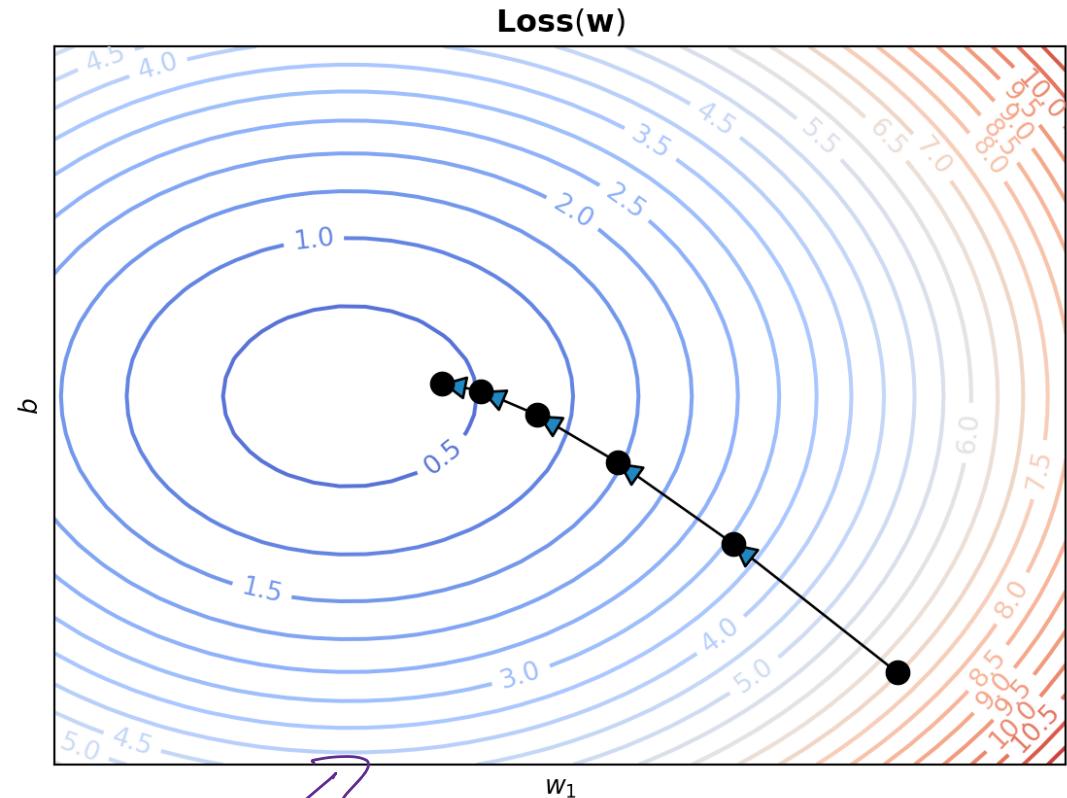
It follows that:

$$\mathbf{w}^* = \mathbf{w}^* - \nabla f(\mathbf{w}^*)$$

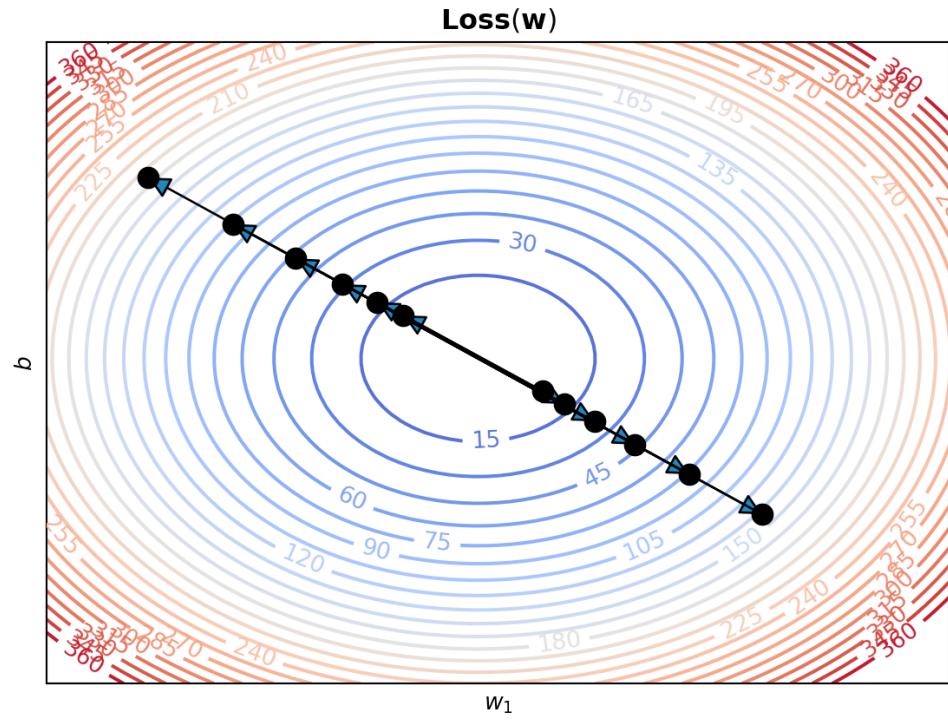
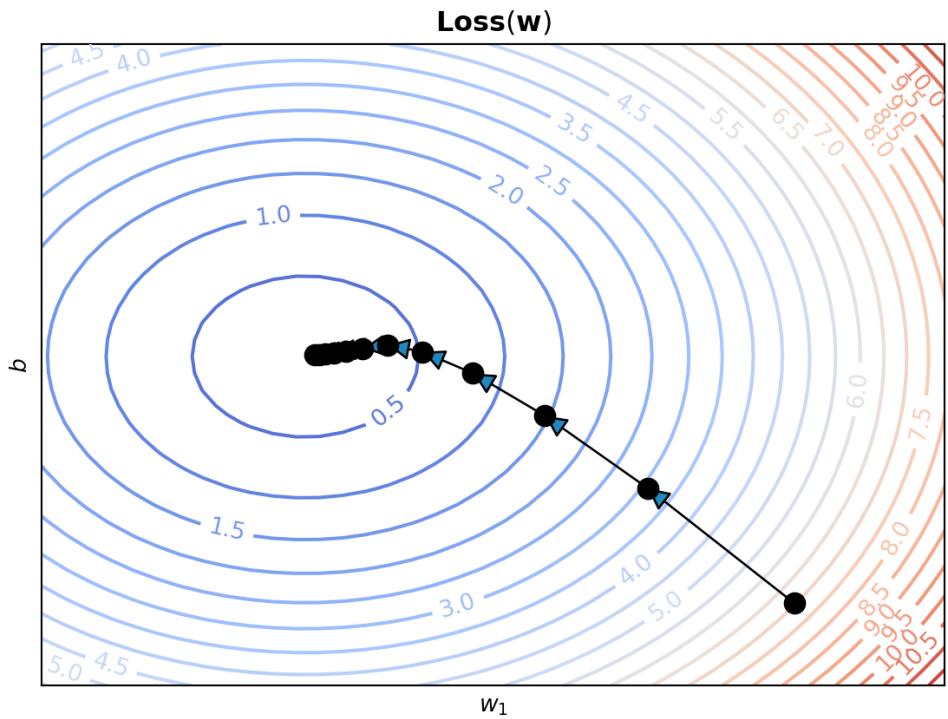
While $\|\nabla f(\mathbf{w}^{(i)})\|_2 > \epsilon$: $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \nabla f(\mathbf{w}^{(i)})$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$$

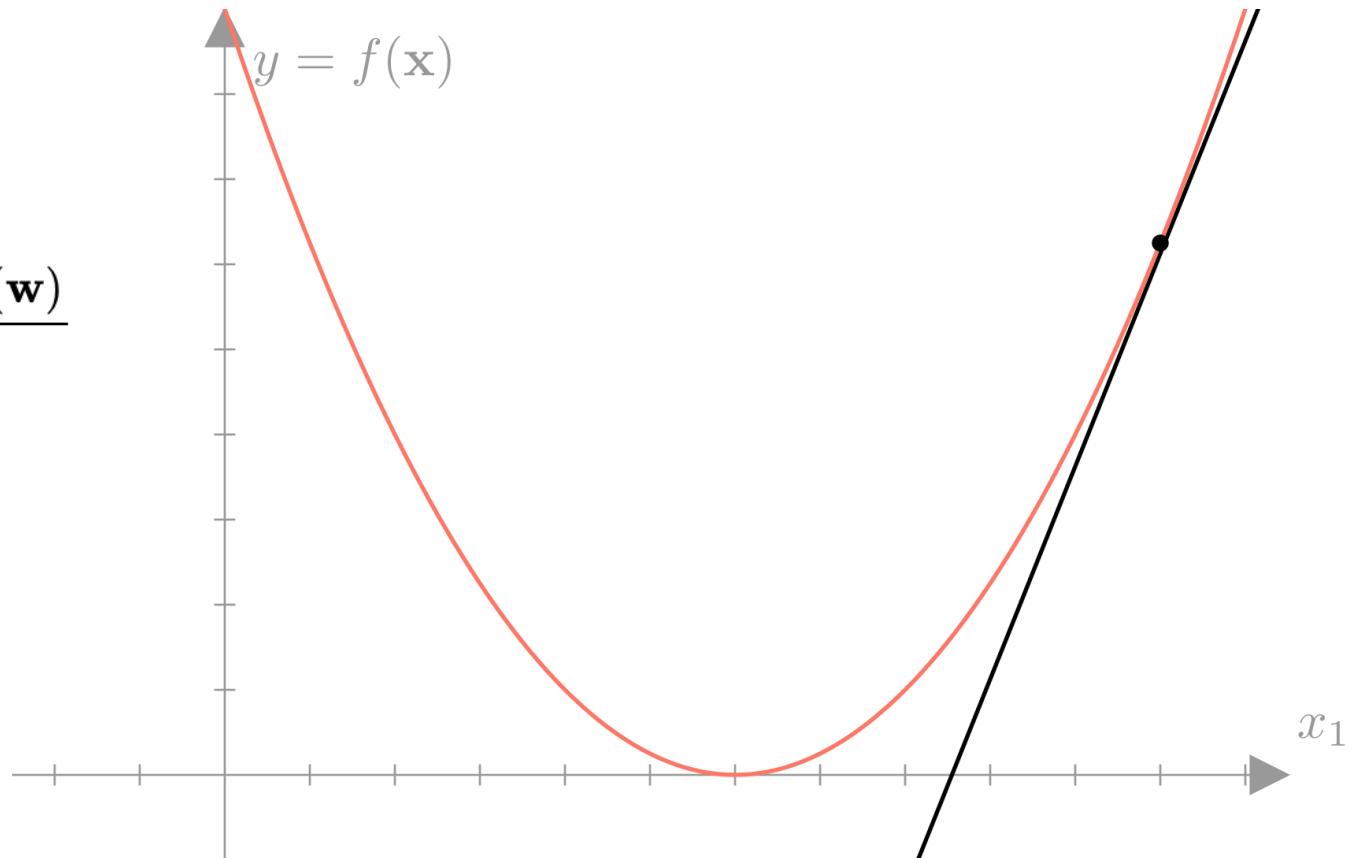
2-norm (magnitude)



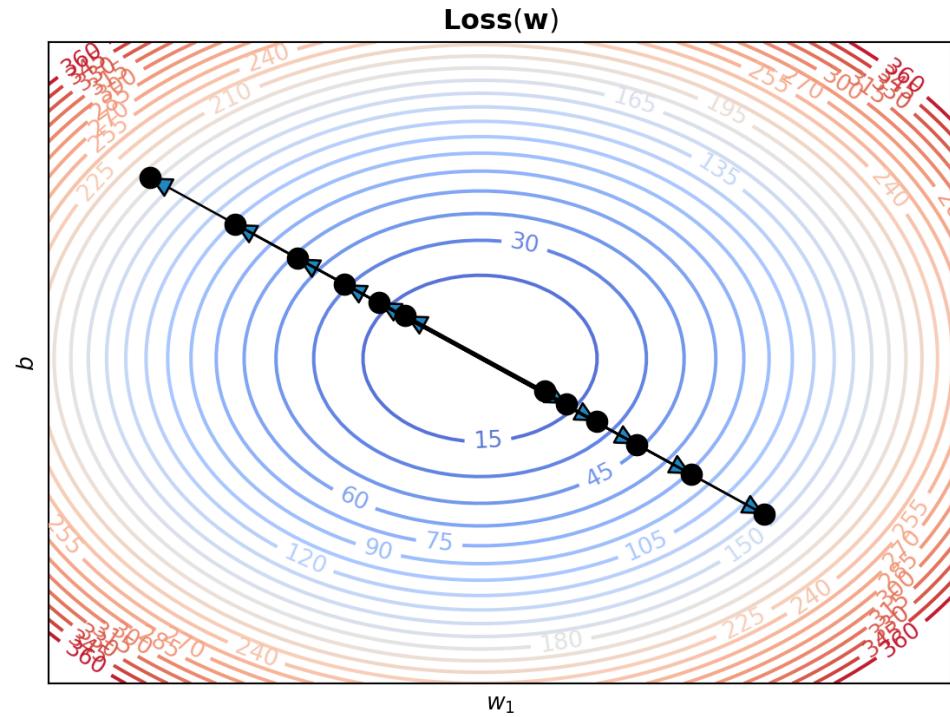
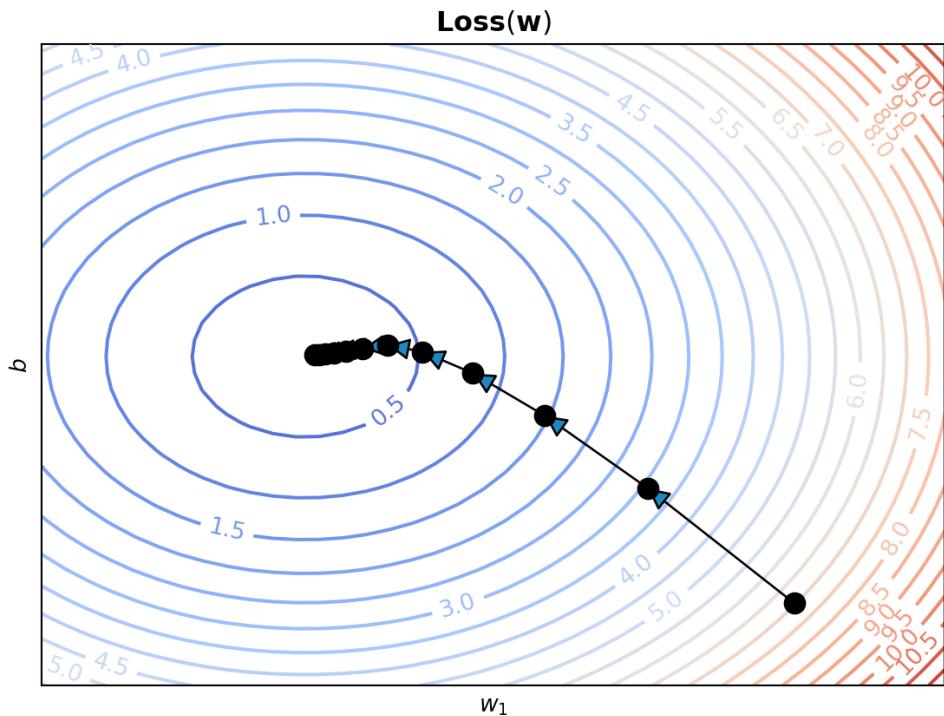
$\|\nabla f(\mathbf{w})\|_2 < \epsilon$ — Stop



$$\frac{df}{d\mathbf{w}} = \lim_{\gamma \rightarrow 0} \max_{\|\epsilon\|_2 < \gamma} \frac{f(\mathbf{w} + \epsilon) - f(\mathbf{w})}{\|\epsilon\|_2}$$



$$\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \alpha \nabla f(\mathbf{w}^{(i)})$$



$$\nabla_{\mathbf{w}} \text{MSE}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{d}{d\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2 \right)$$

$$= \frac{2}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i) \mathbf{x}_i$$

With this gradient our gradient descent update becomes:

$$\mathbf{w}^{(i+1)} \longleftarrow \mathbf{w}^{(i)} - \alpha \left(\frac{2}{N} \right) \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w}^{(i)} - y_i) \mathbf{x}_i$$

$$\nabla_{\mathbf{w}} \text{MSE}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{d}{d\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2 \right)$$

$$= \frac{2}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i) \mathbf{x}_i$$

We know that at the minimum, the gradient must be $\mathbf{0}$, so the following condition must hold:

$$\mathbf{0} = \left(\frac{2}{N} \right) \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i) \mathbf{x}_i$$

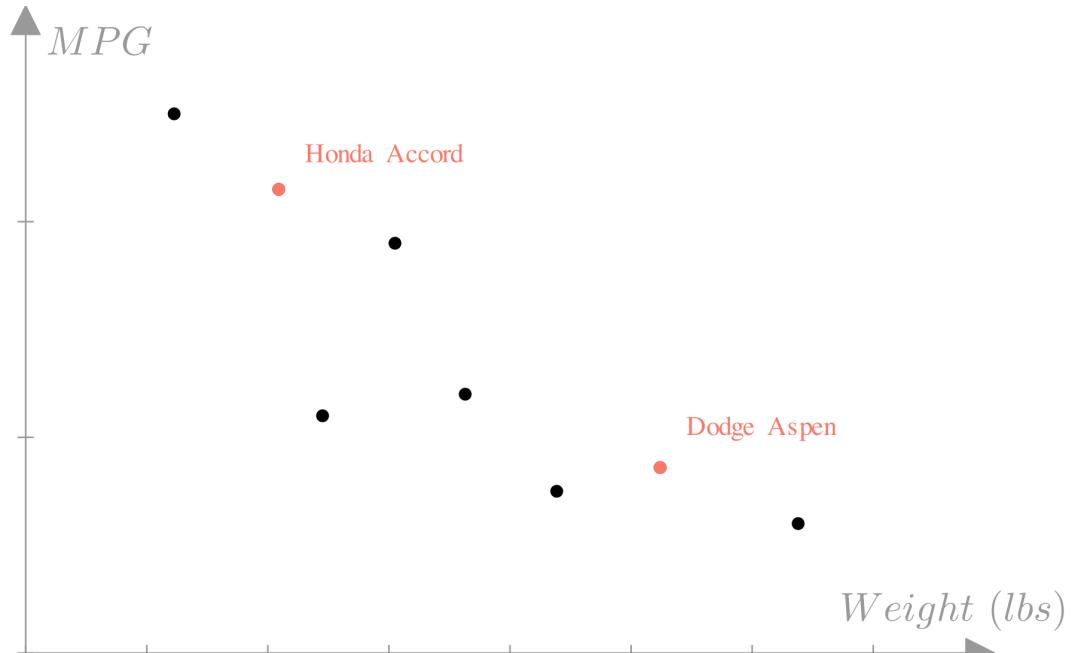
$$\left(\sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left(\sum_{i=1}^N y_i \mathbf{x}_i \right) = \mathbf{w}^*$$

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{y} \mathbf{X})$$

Input: $\mathbf{x}_i = \begin{bmatrix} \text{Weight} \\ \text{Horsepower} \\ \text{Displacement} \\ 0\text{-60mph} \end{bmatrix}$, Output: $y_i = MPG$

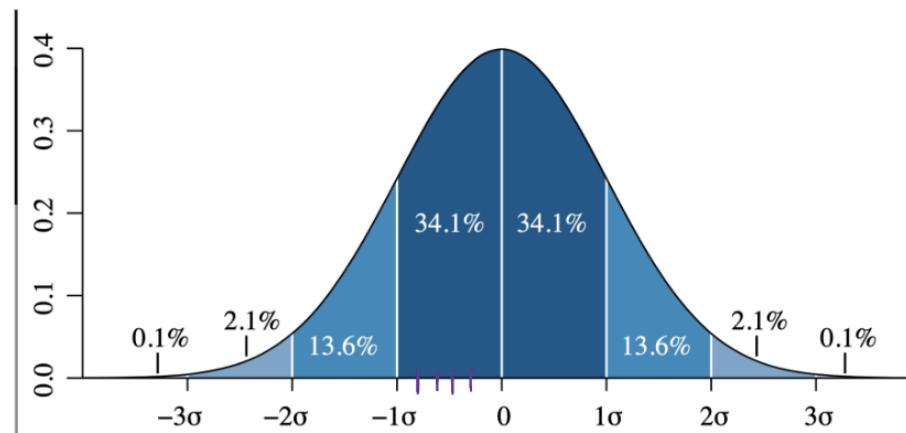
Predicted MPG = $f(\mathbf{x}) =$

$$(\text{weight})w_1 + (\text{horsepower})w_2 + (\text{displacement})w_3 + (0\text{-60mph})w_4$$



$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right)$$

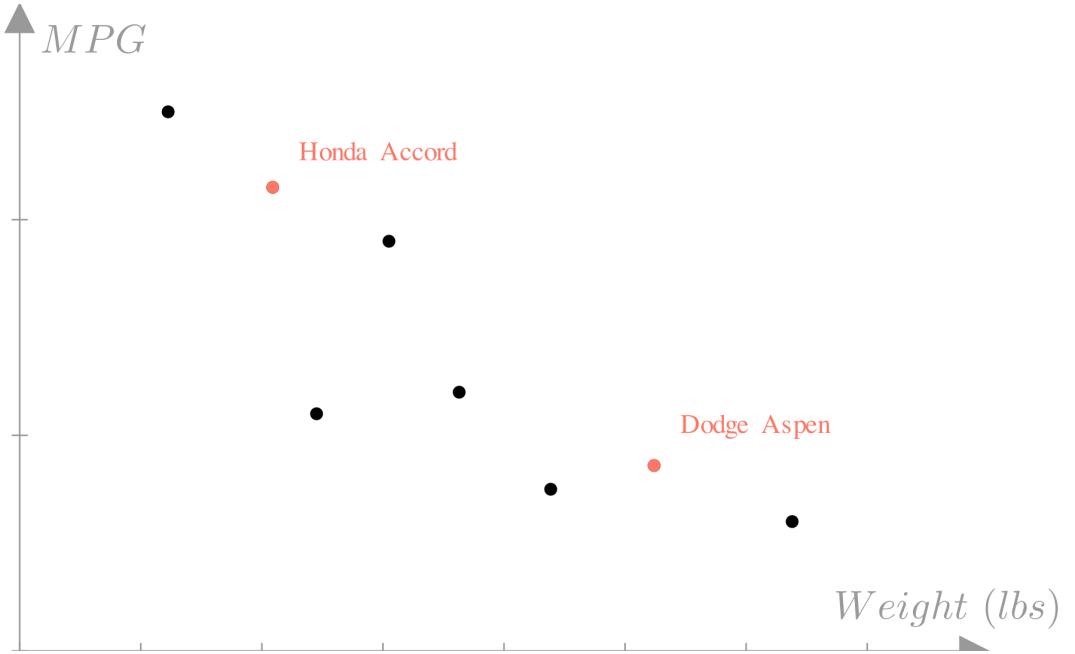
Normal distribution



$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right)$$

$$y_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma^2)$$

$$p(y_i | \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T \mathbf{w})^2\right)$$

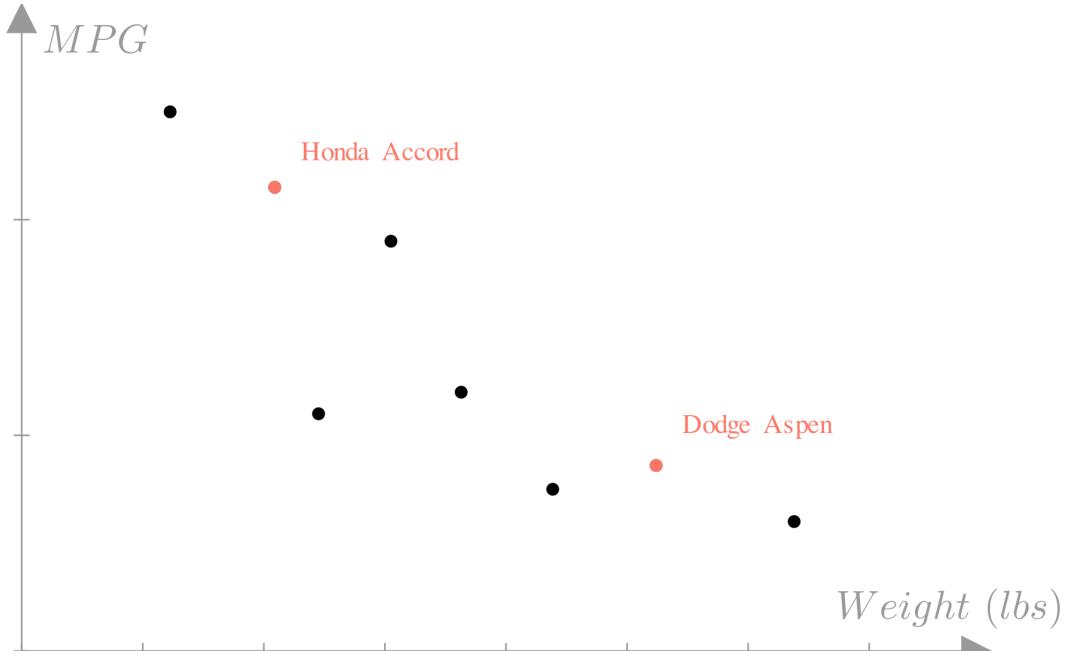


$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right)$$

$$y_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma^2)$$

$$p(y_i | \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T \mathbf{w})^2\right)$$

$$e_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w} - y_i, \sigma^2)$$



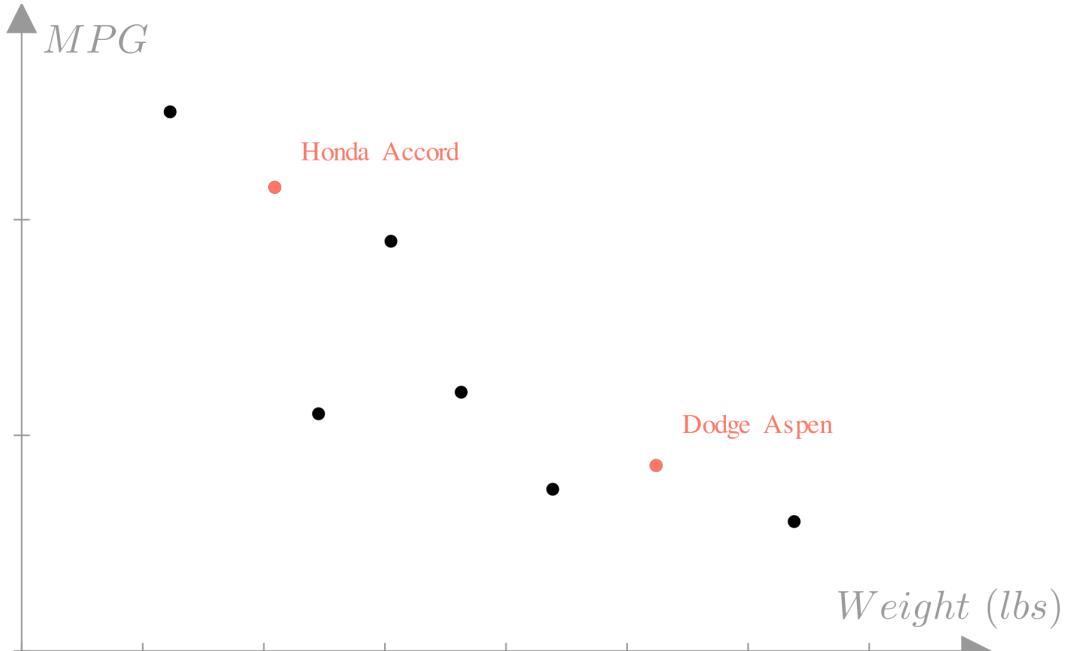
$$p(e_i | \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T \mathbf{w})^2\right)$$

$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y - \mu)^2\right)$$

$$y_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma^2)$$

$$p(y_i | \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T \mathbf{w})^2\right)$$

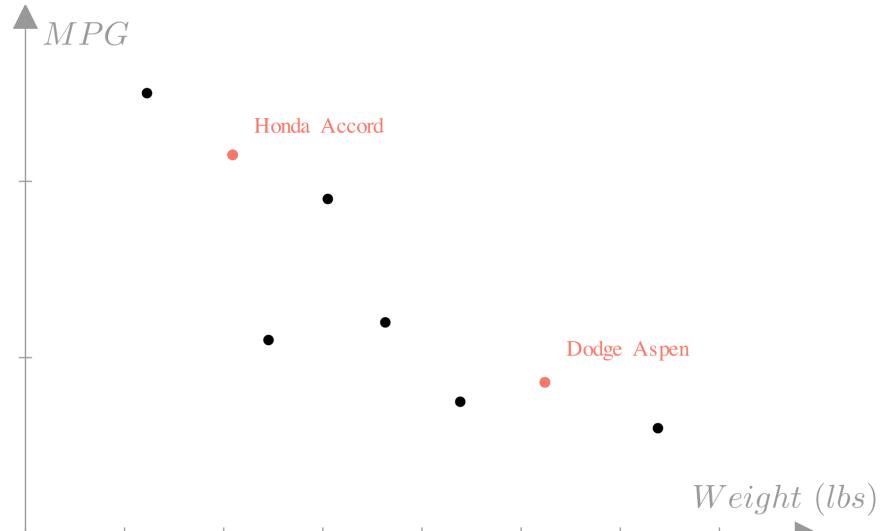
$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} p(\mathbf{y} | \mathbf{X}, \mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} p(y_1, \dots, y_N | \mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{w})$$



$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} p(y_1, \dots, y_N \mid \mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{w})$$

$$p(y_i \mid \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y_i - \mathbf{x}_i^T \mathbf{w})^2\right)$$

$$p(y_1, \dots, y_N \mid \mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{w}) = \prod_{i=1}^N p(y_i \mid \mathbf{x}_i, \mathbf{w})$$



$$\underset{\mathbf{w}}{\operatorname{argmax}} \prod_{i=1}^N p(y_i \mid \mathbf{x}_i, \mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} - \sum_{i=1}^N \log p(y_i \mid \mathbf{x}_i, \mathbf{w}) = \mathbf{NLL}(\mathbf{w}, \mathbf{X}, \mathbf{y})$$

$$\textbf{NLL}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = -\sum_{i=1}^N \log \left[\frac{1}{\sigma \sqrt{2\pi}} \exp \left(-\frac{1}{2\sigma^2} (y_i - \mathbf{x}_i^T \mathbf{w})^2 \right) \right]$$

$$\nabla_{\mathbf{w}} \textbf{NLL}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{d}{d\mathbf{w}} \bigg(\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - \mathbf{x}_i^T \mathbf{w})^2 + N \log \sigma \sqrt{2\pi} \bigg)$$

$$\underset{\mathbf{w}}{\operatorname{argmin}} \ MSE(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \underset{\mathbf{w}}{\operatorname{argmin}} \ \mathbf{NLL}(\mathbf{w}, \mathbf{X}, \mathbf{y})$$