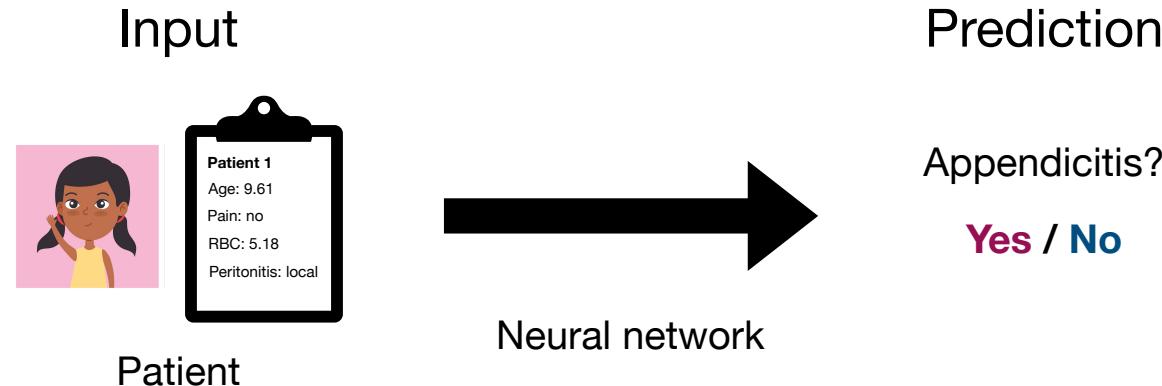


Linear regression, gradient descent & maximum likelihood

Setup

Example: Appendicitis diagnosis



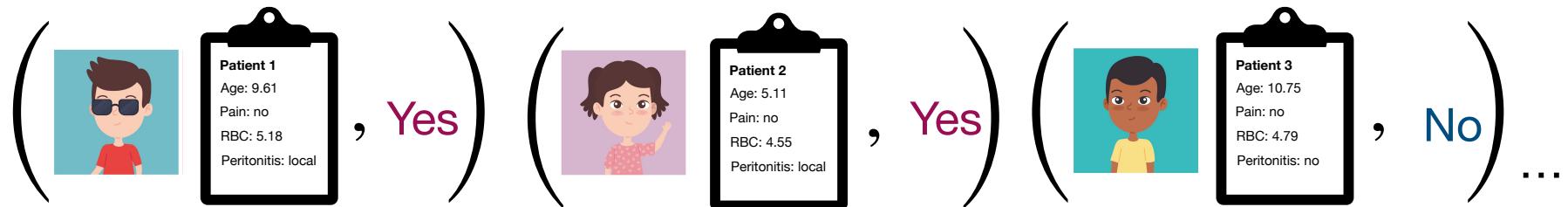
Some notation:

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

Prediction function

Dataset

Set of known inputs and **outputs**

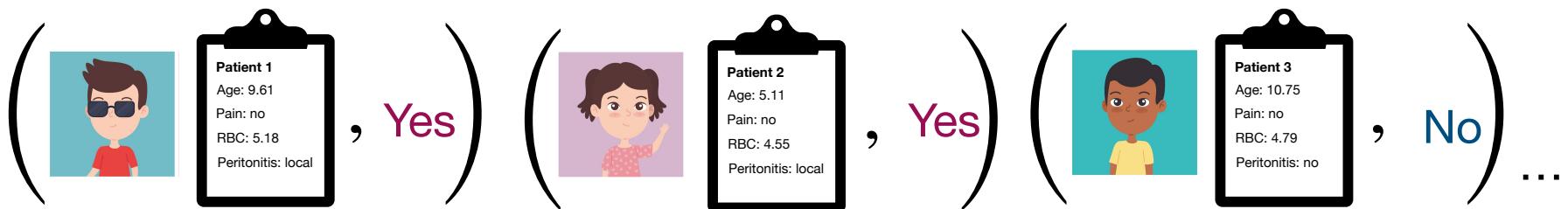


Some notation:

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

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

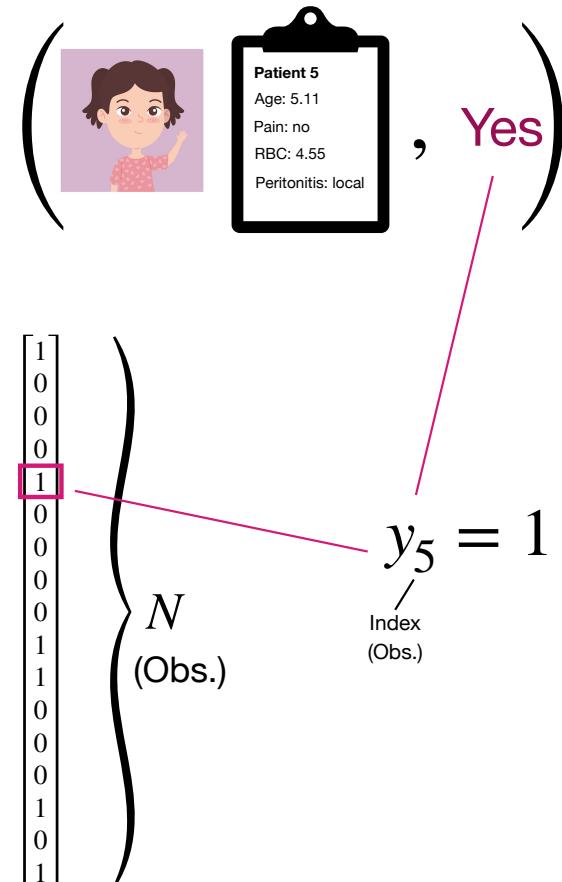
Diagnosis	
	Diagnosis
0	appendicitis
1	no appendicitis
2	no appendicitis
3	no appendicitis
4	appendicitis
5	no appendicitis
6	no appendicitis
7	no appendicitis
8	no appendicitis
9	appendicitis
10	appendicitis
11	no appendicitis
12	no appendicitis
13	no appendicitis
14	appendicitis
15	no appendicitis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

Output table

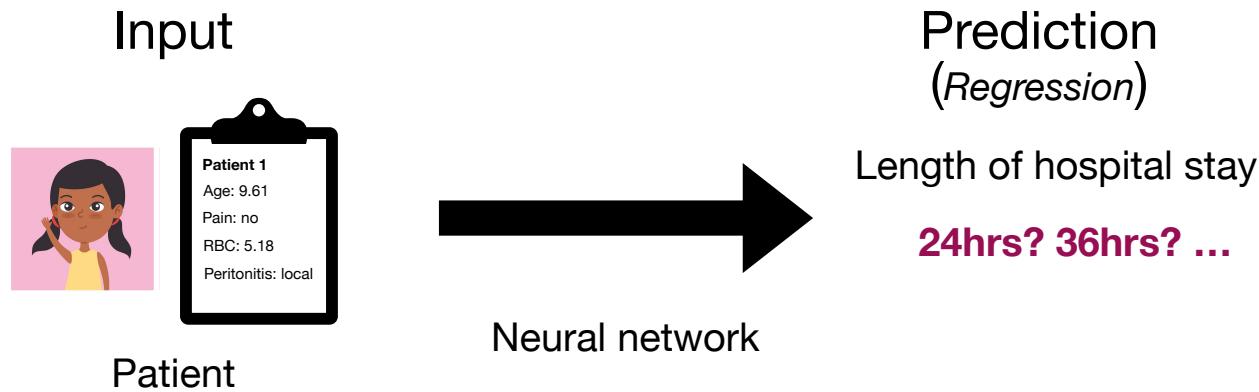
Converted

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

$y =$



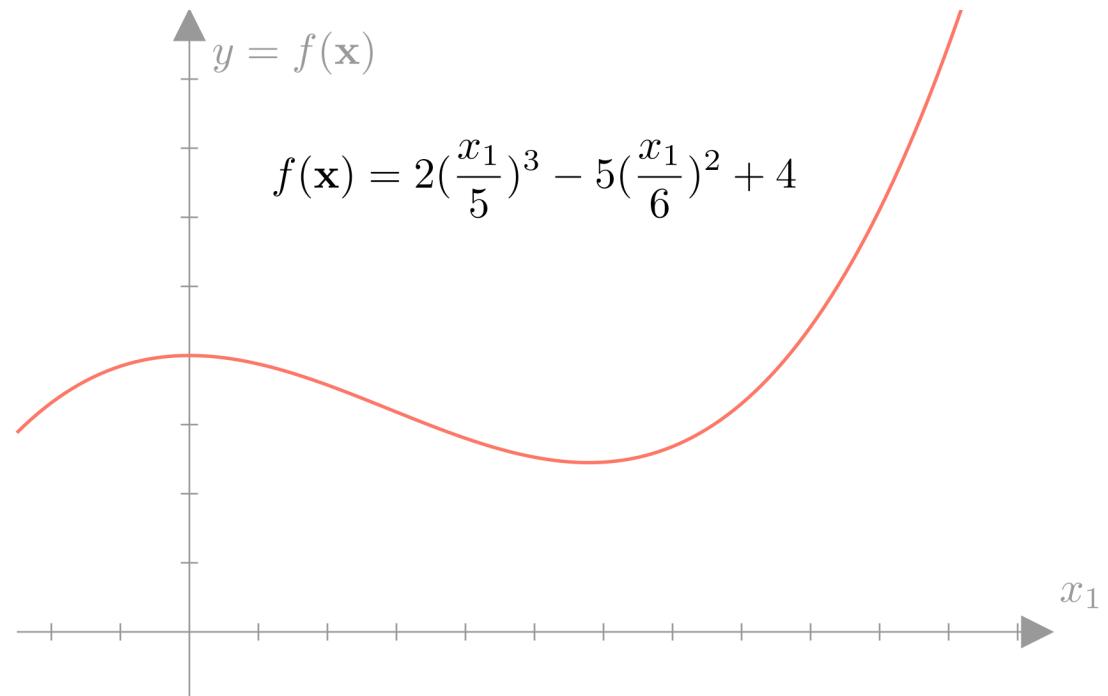
Quantitative outputs: *Regression*



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

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

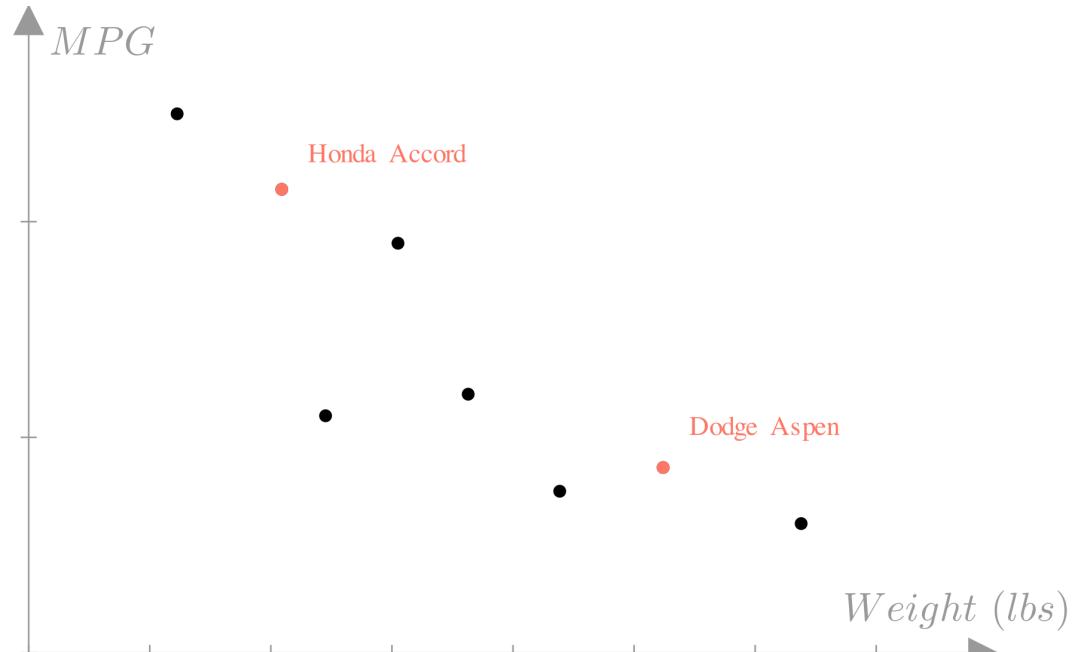
Input: $\mathbf{x} \xrightarrow{predict}$ Output: y , $y = f(\mathbf{x})$
Prediction function



Honda Accord: $\begin{bmatrix} \text{Weight:} & 2500 \text{ lbs} \\ \text{Horsepower:} & 123 \text{ HP} \\ \text{Displacement:} & 2.4 \text{ L} \\ \text{0-60mph:} & 7.8 \text{ Sec} \end{bmatrix} \rightarrow \text{MPG: } 33\text{mpg}$

Dodge Aspen: $\begin{bmatrix} \text{Weight:} & 3800 \text{ lbs} \\ \text{Horsepower:} & 155 \text{ HP} \\ \text{Displacement:} & 3.2 \text{ L} \\ \text{0-60mph:} & 6.8 \text{ Sec} \end{bmatrix} \rightarrow \text{MPG: } 21\text{mpg}$

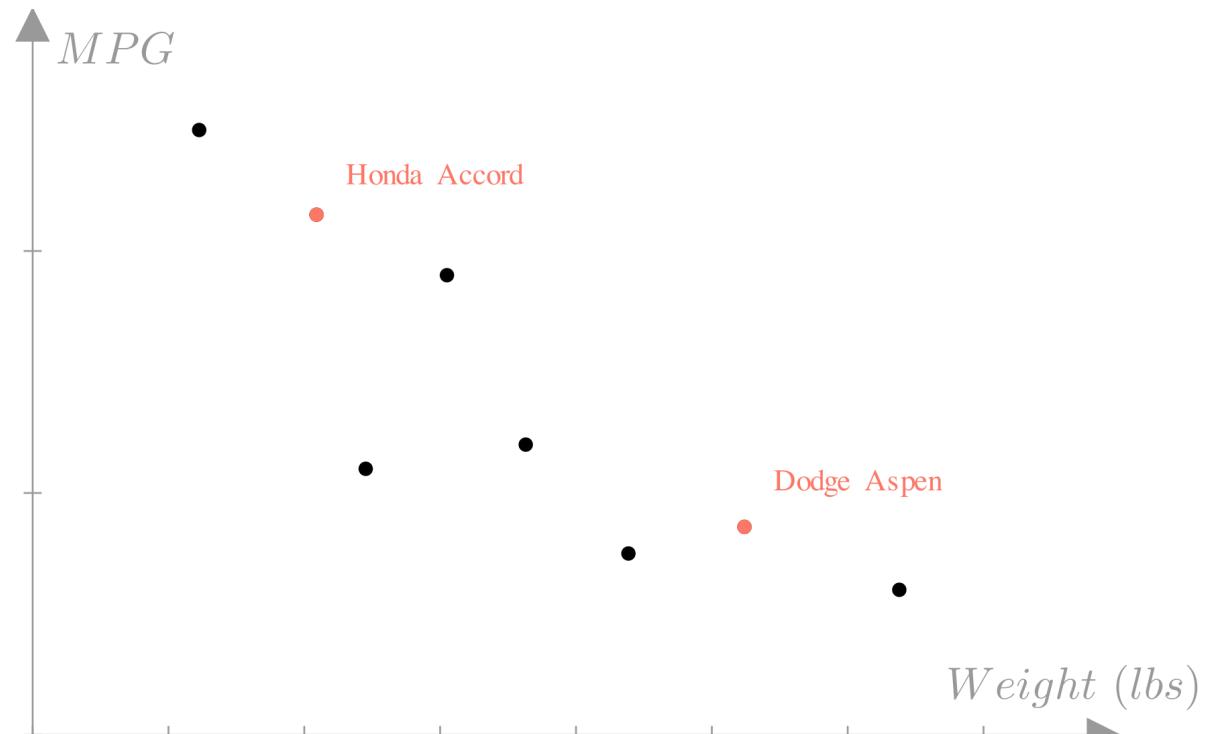
: :

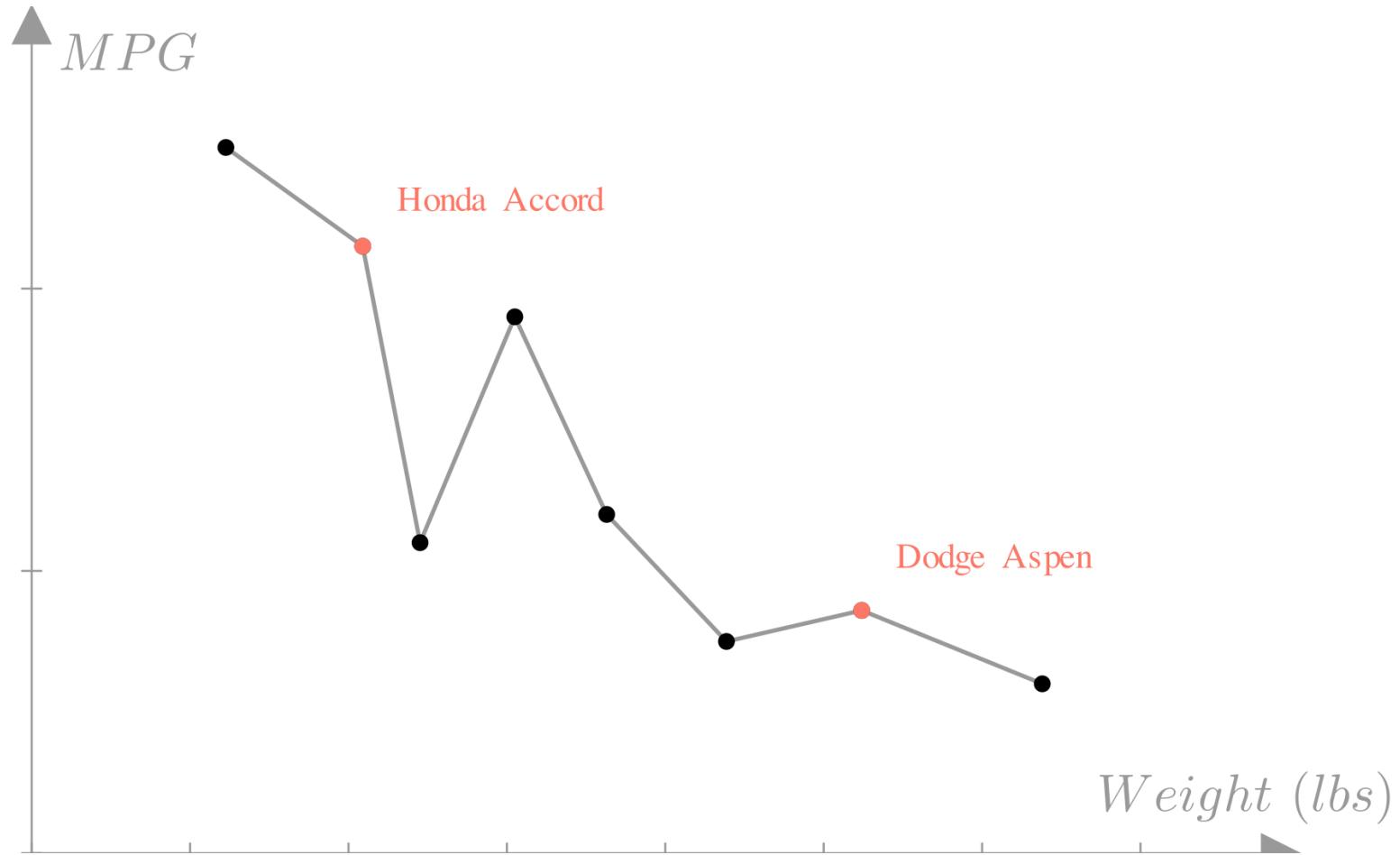


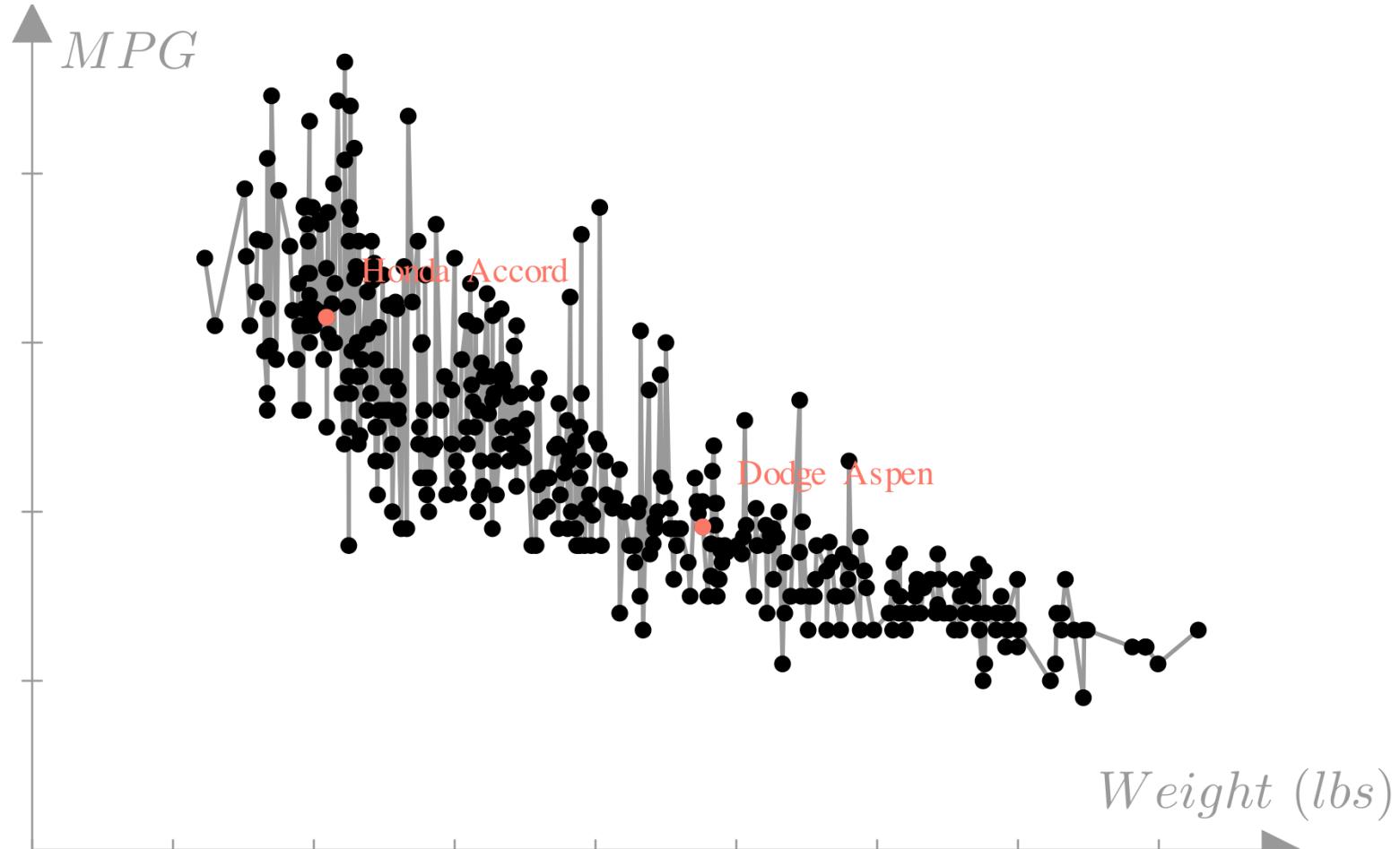
Dataset: $\mathcal{D} = \{(\mathbf{x}_i, y_i) \text{ for } i \in 1 \dots N\}$

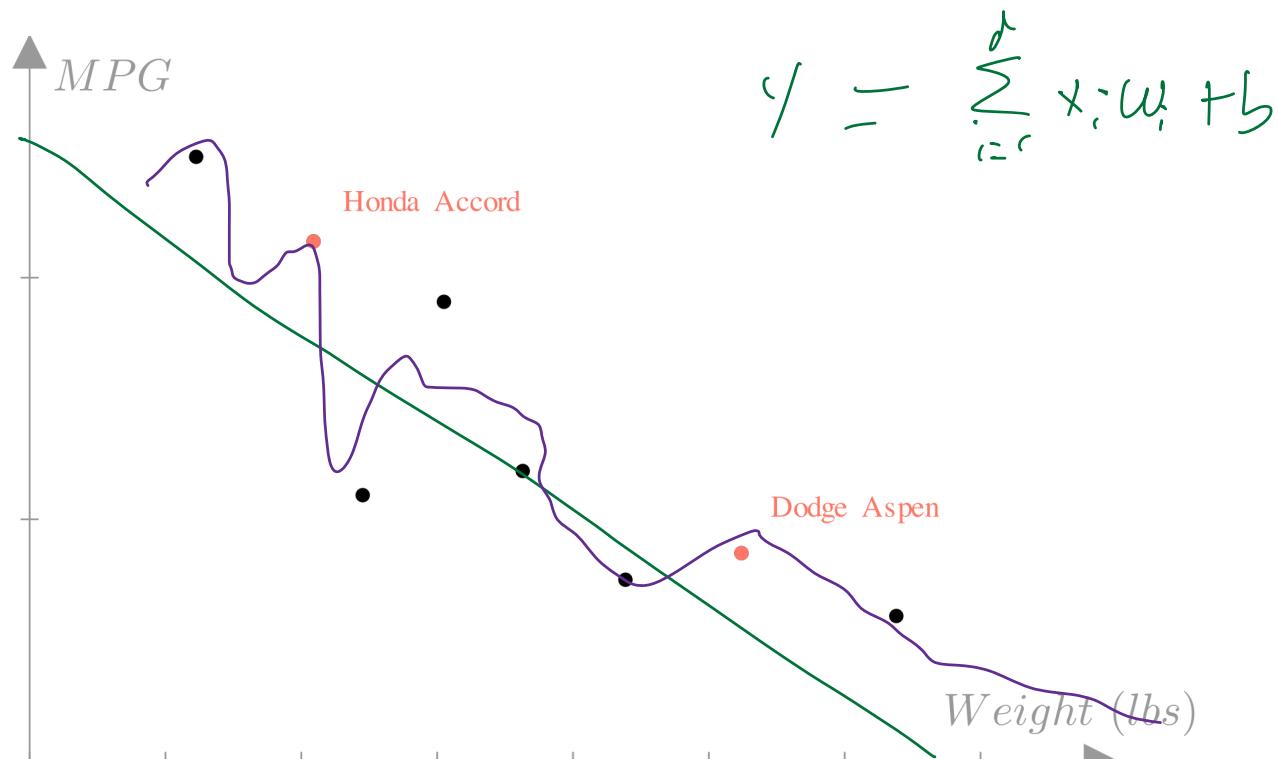
$$\text{mpg} = f(\text{weight}, \text{horsepower} \dots)$$

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









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

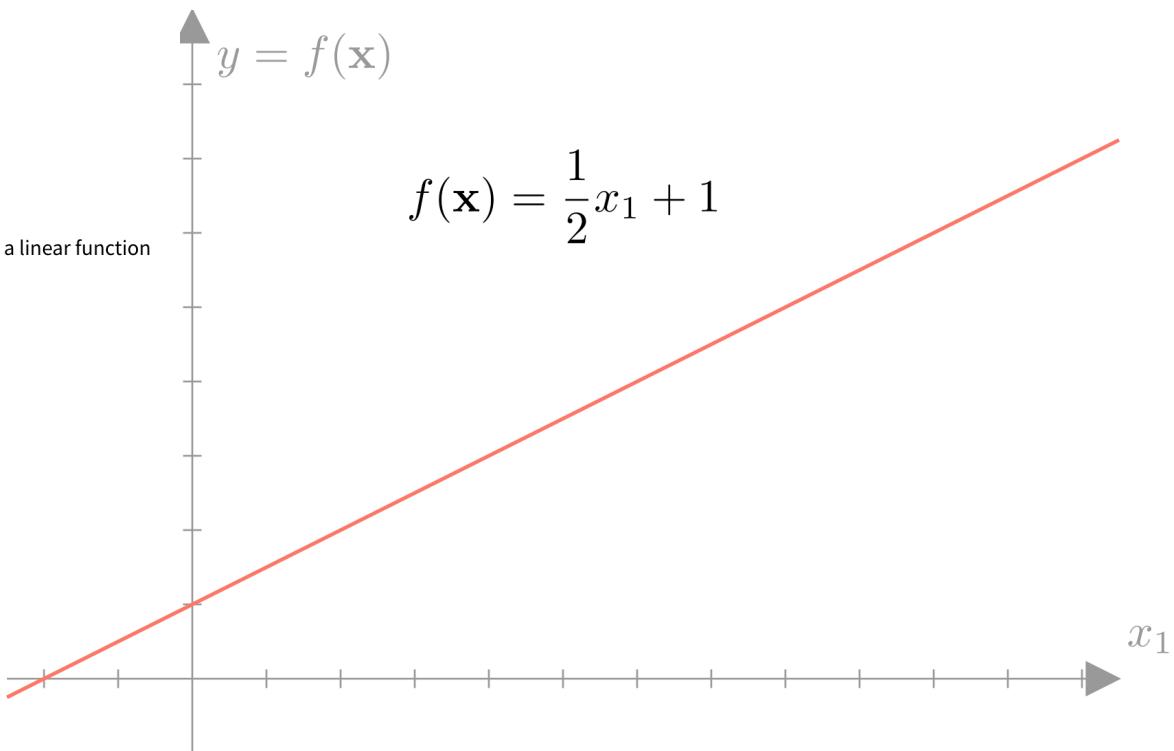
$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

and

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

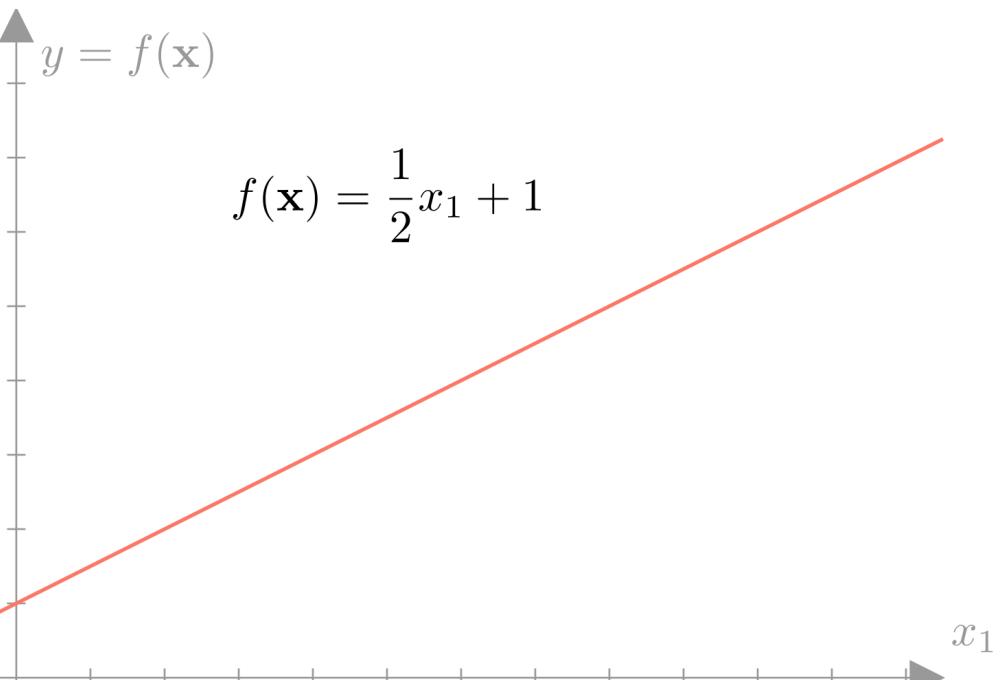
For a linear function, the output can be defined as a weighted sum of the inputs. In other words a linear function of a vector can always be written as:

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



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

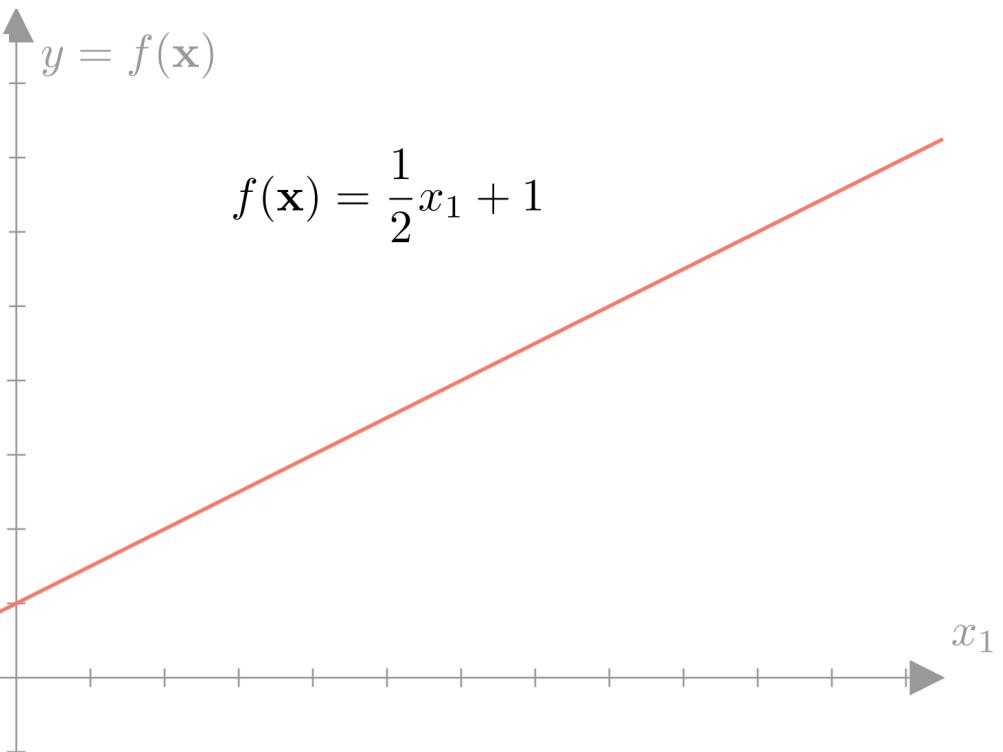
$$f(\mathbf{x}) = \sum_{i=1}^n x_i w_i + b$$



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

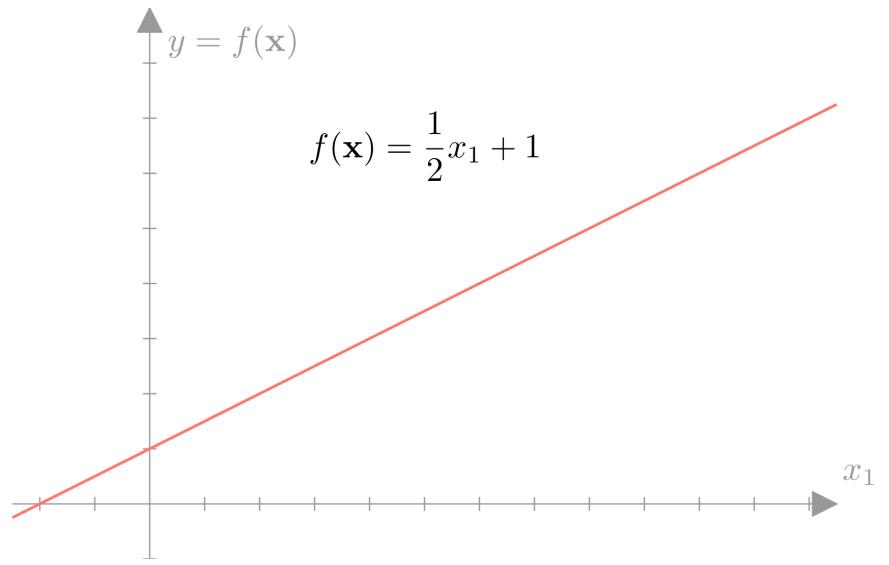
```
def f(x):
    w = np.array([-0.6, -0.2])
    b = -1
    return np.dot(x, w) + b
```

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

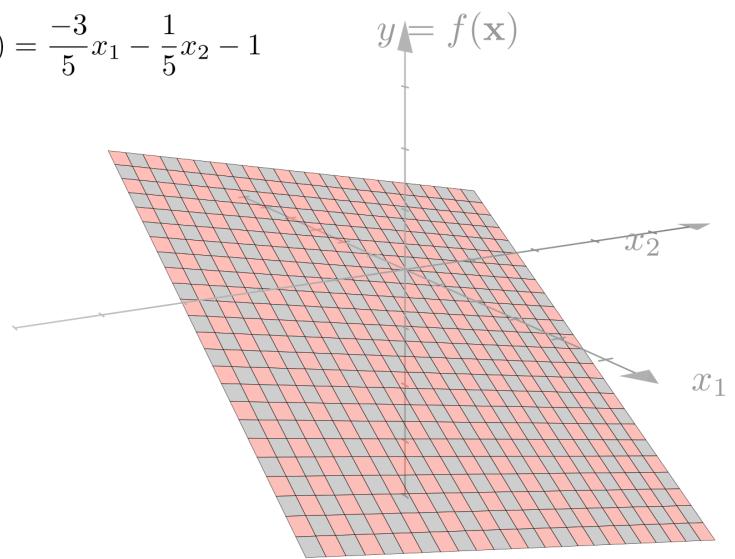


We can easily see then that using this notation:

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



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



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:

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

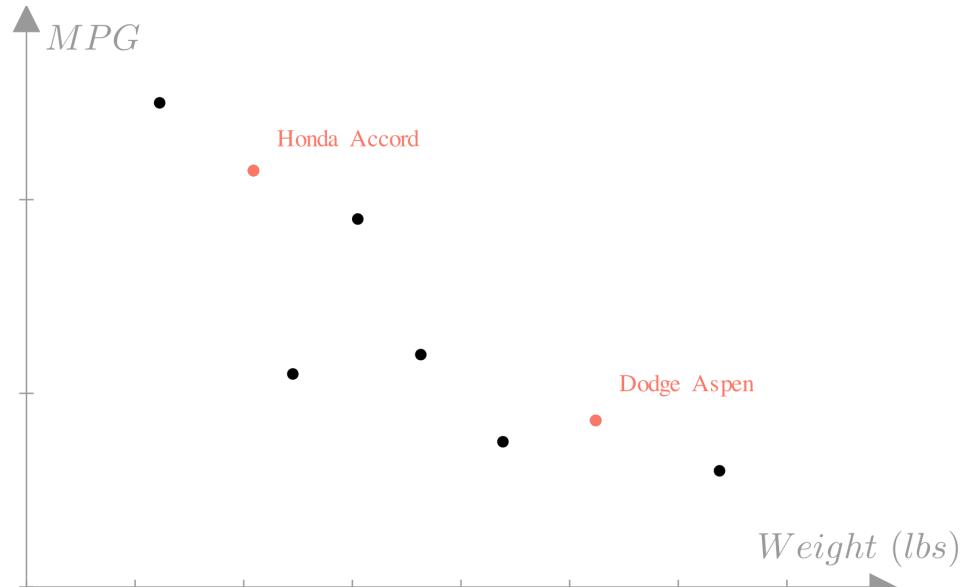
Meaning that the output will be a weighted sum of the *features* of the input. In the case of our car example, we will make predictions as:

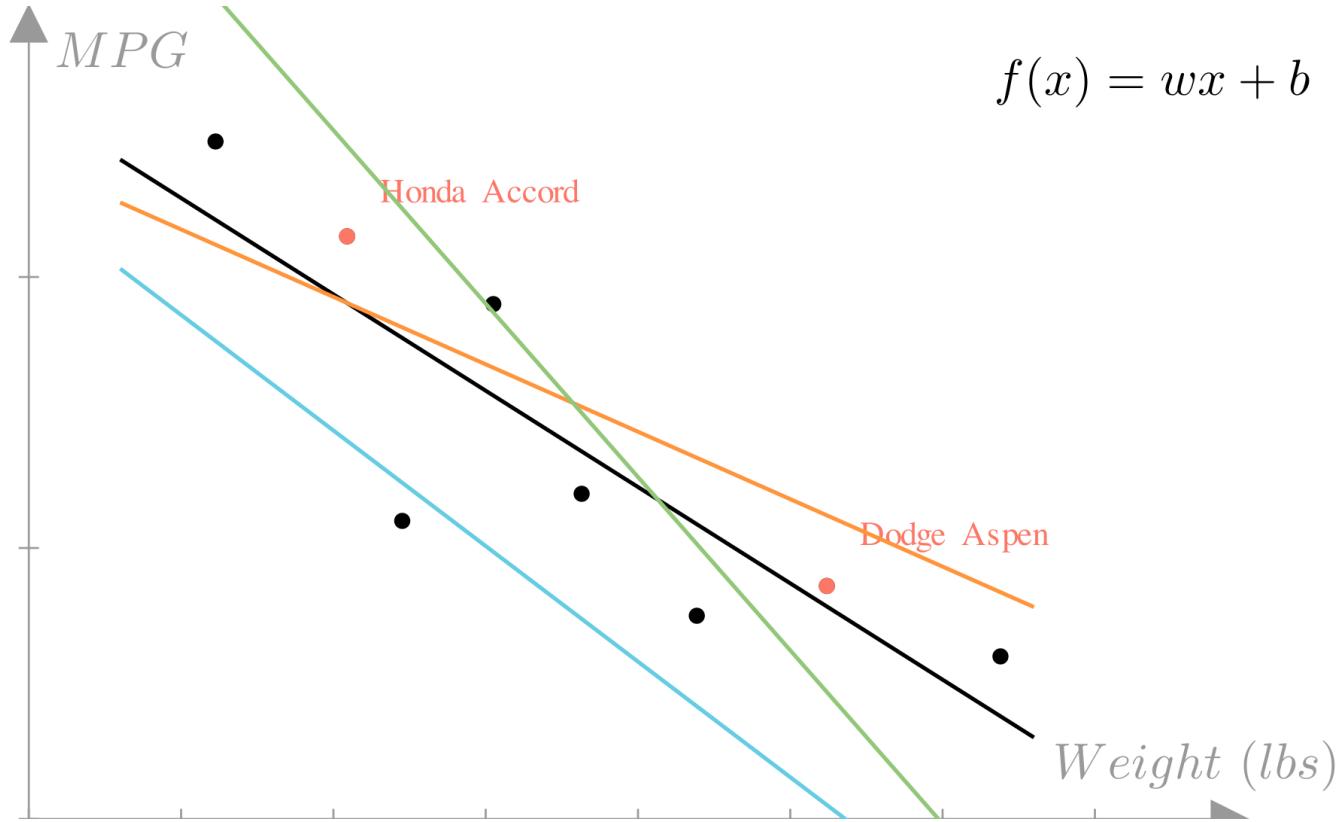
$$\text{Predicted MPG} = f(\mathbf{x}) =$$

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

Or in matrix notation:

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





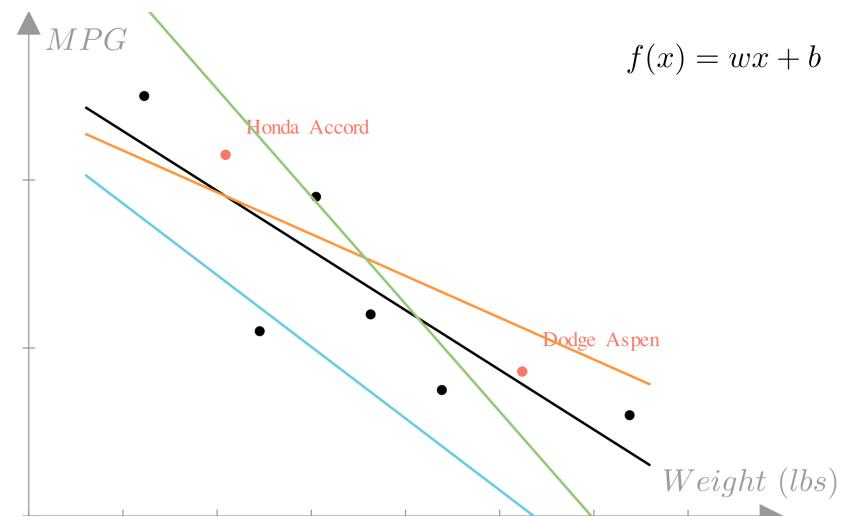
$$f(\mathbf{x}) = \sum_{i=1}^n x_i w_i + b$$

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)

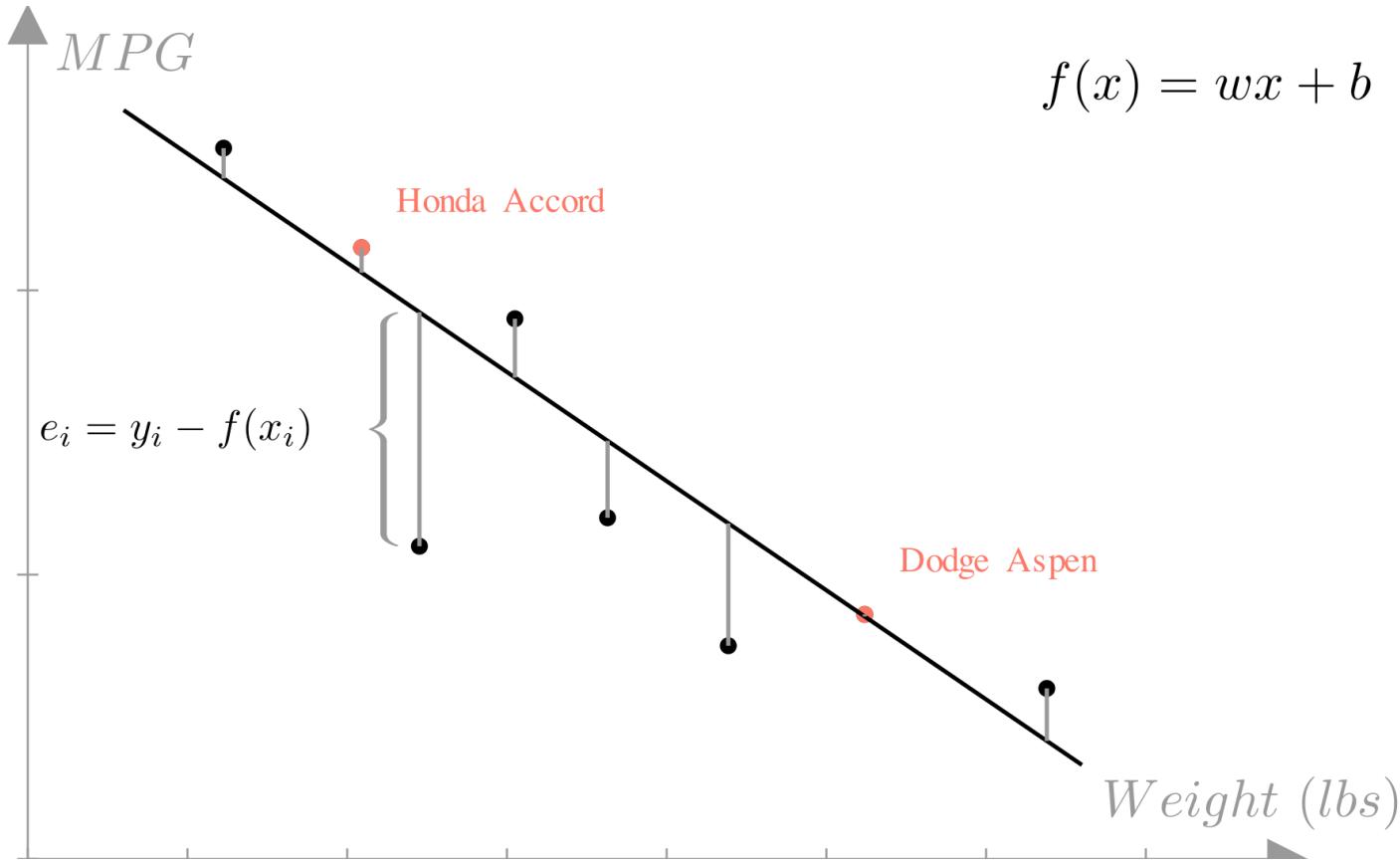
```
class Regression:
    def __init__(self, weights):
        self.weights = weights

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



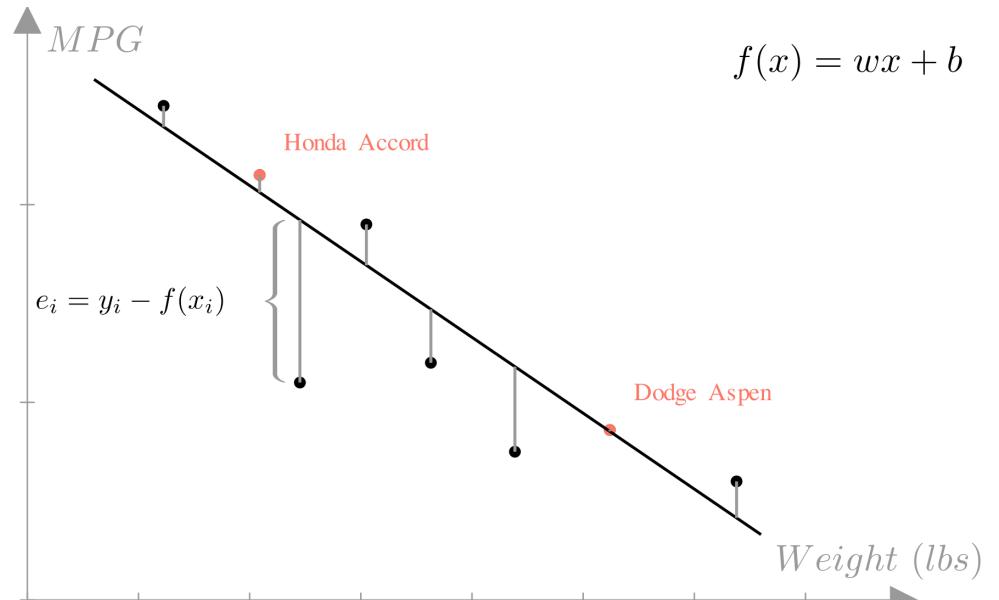
The **residual** or **error** of a prediction is the difference between the prediction and the true output:

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



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

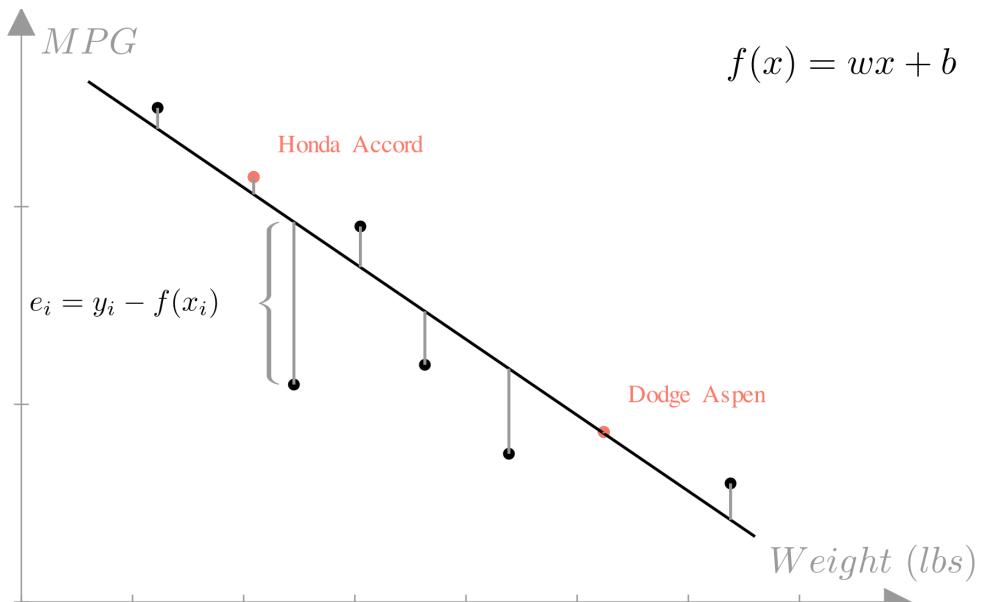
$$MSE = \frac{1}{N} \sum_{i=1}^N (f(\mathbf{x}_i) - y_i)^2 = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$



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

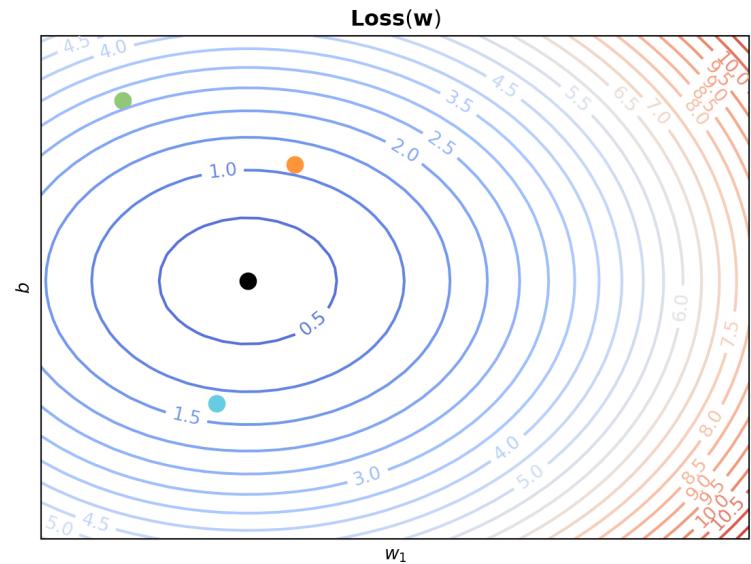
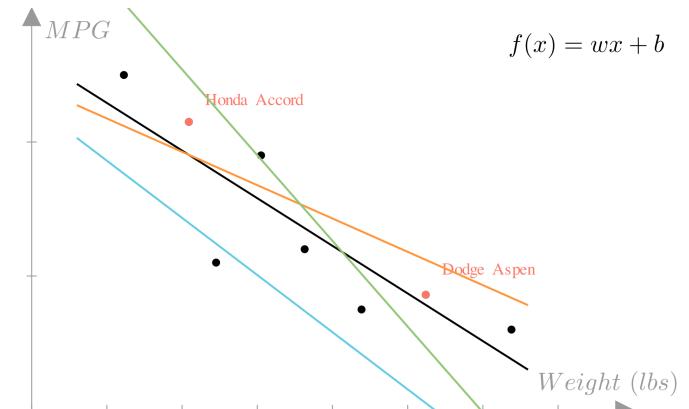
$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

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



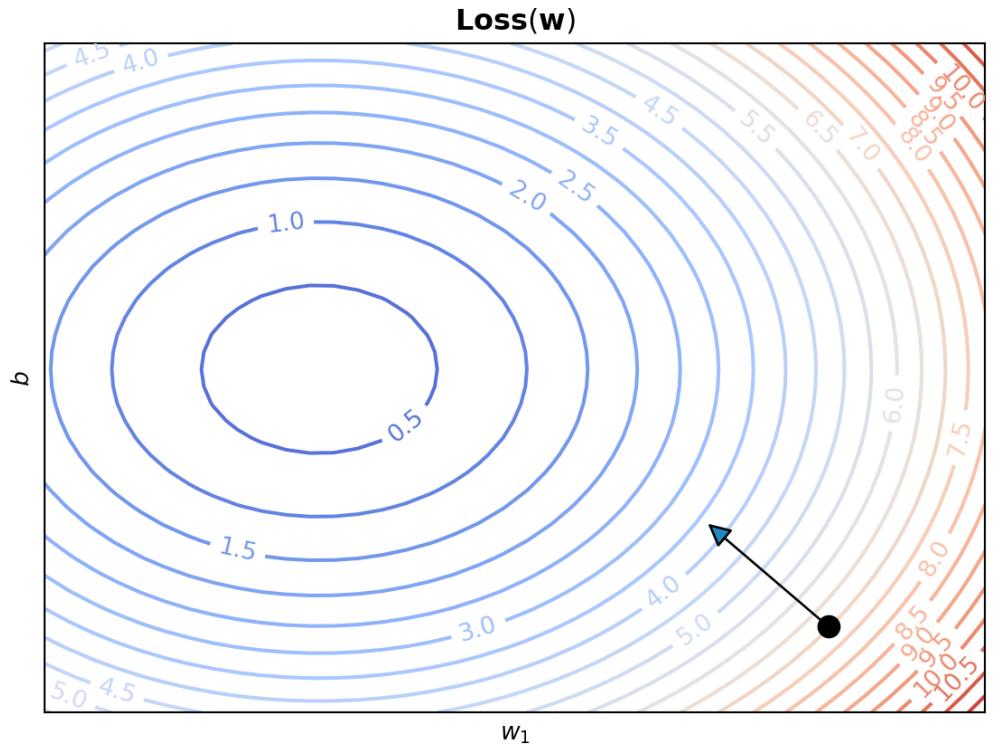
$$\text{Loss}(\mathbf{w}) = MSE(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

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



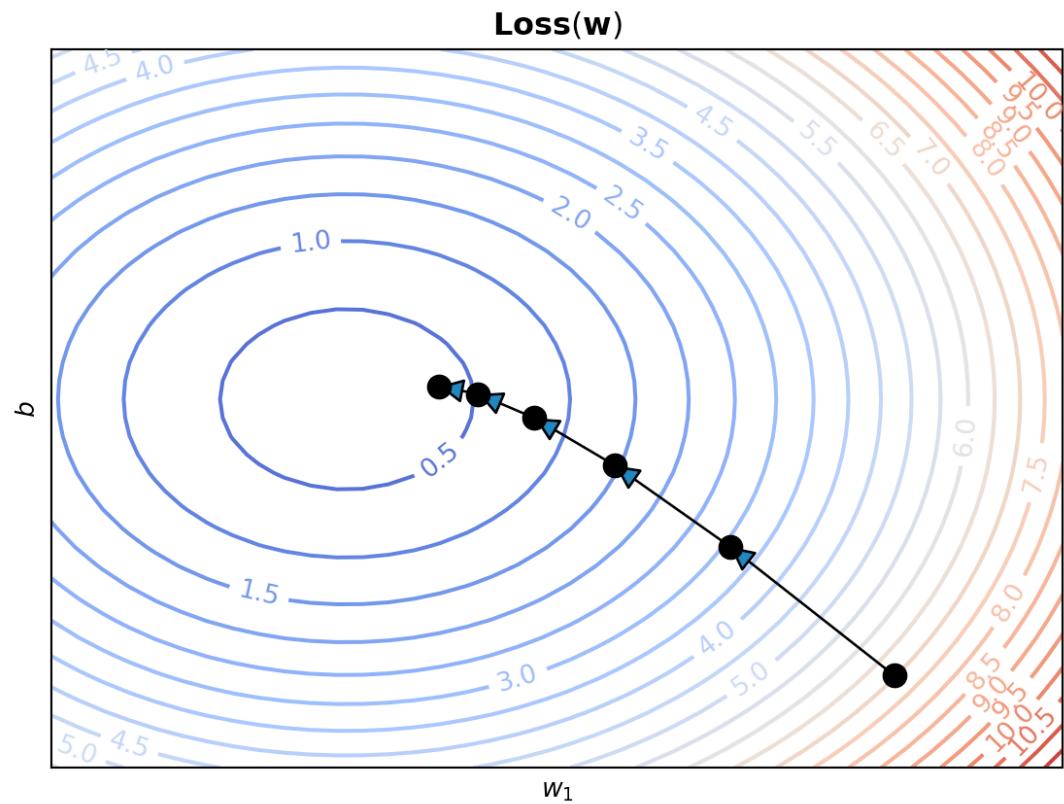
Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

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



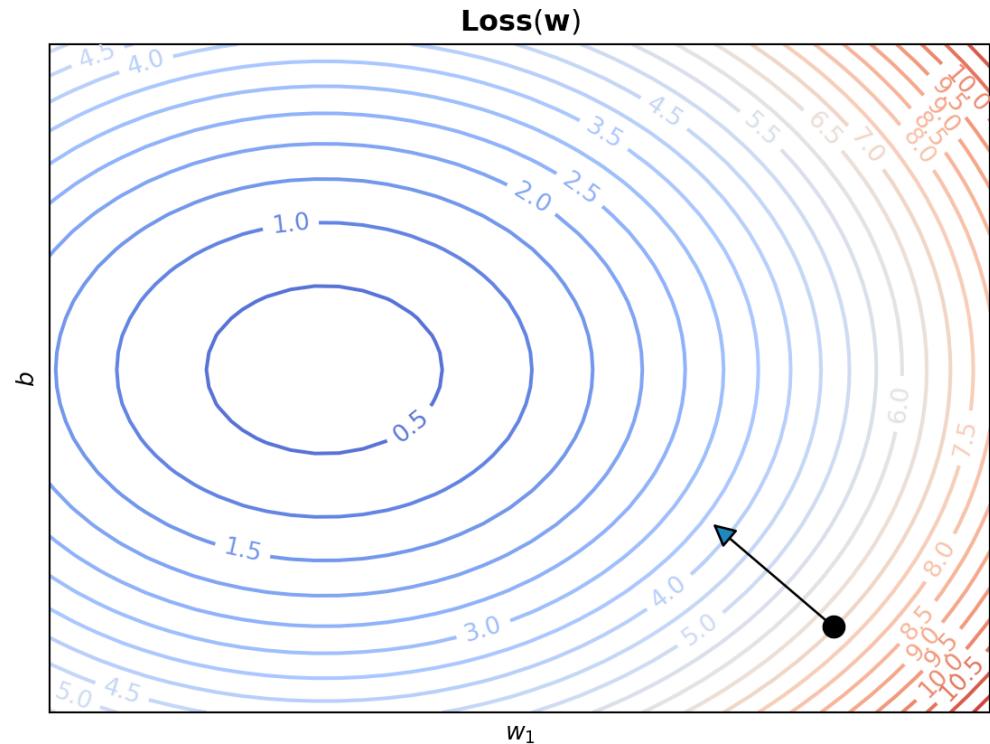
Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

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



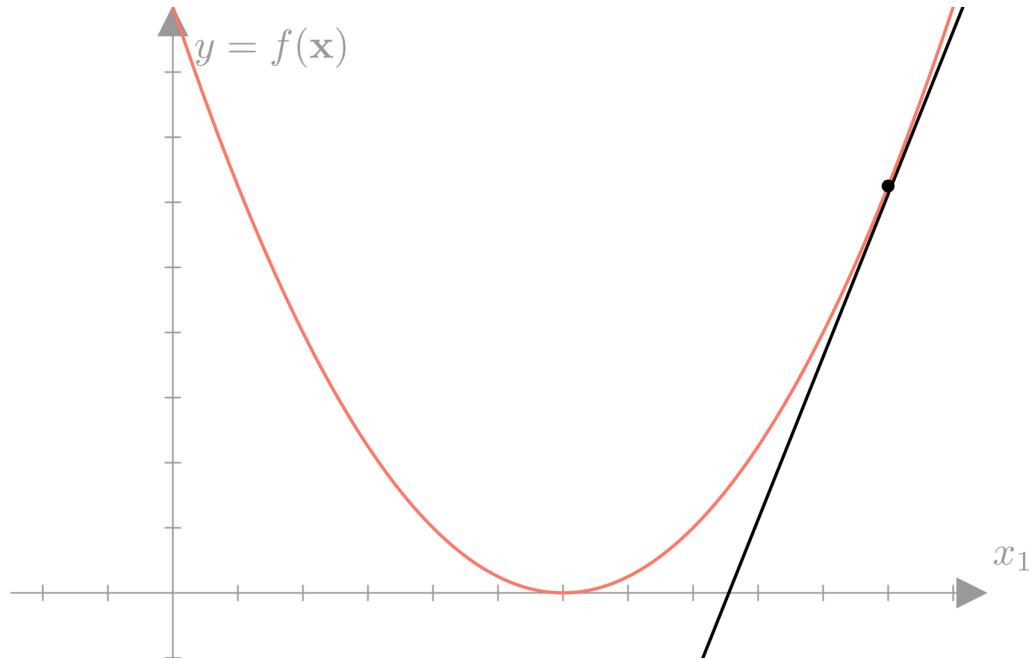
Find: $\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} f(\mathbf{w})$

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



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.

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

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

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.

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

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

$$\frac{d}{d\mathbf{x}} \mathbf{x}^T \mathbf{x} =$$

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.

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

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

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

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.

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

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

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

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.

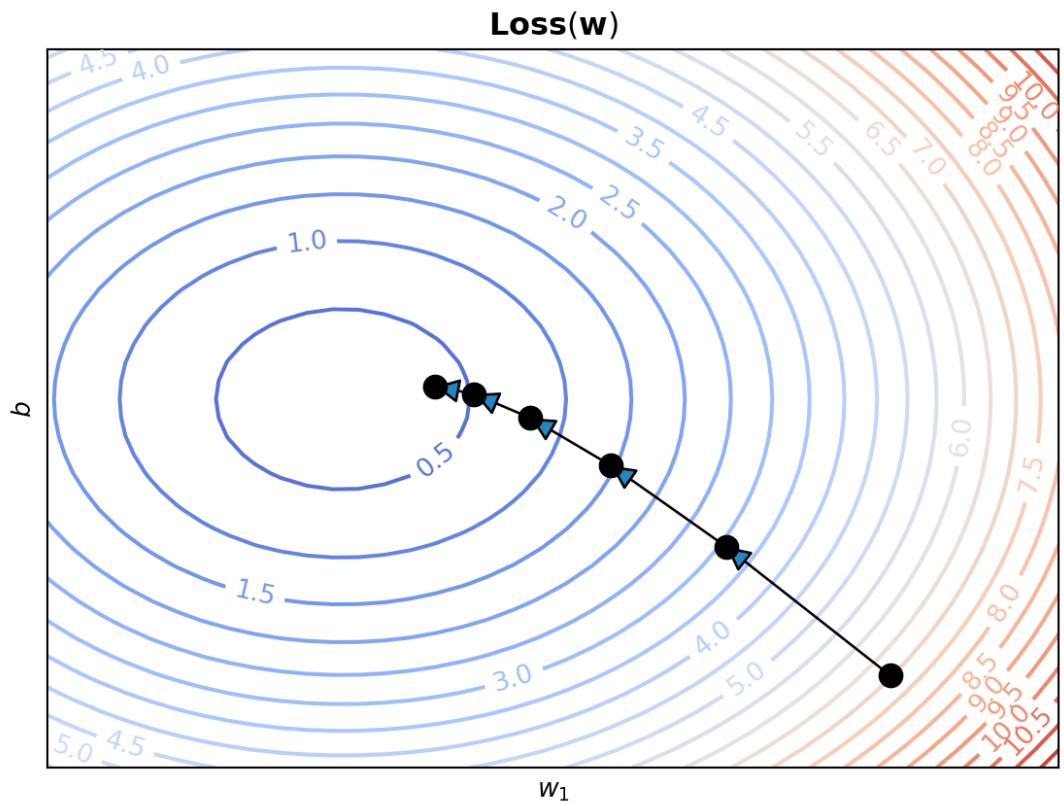
$$\frac{df}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \\ \vdots \end{bmatrix}$$

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


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

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

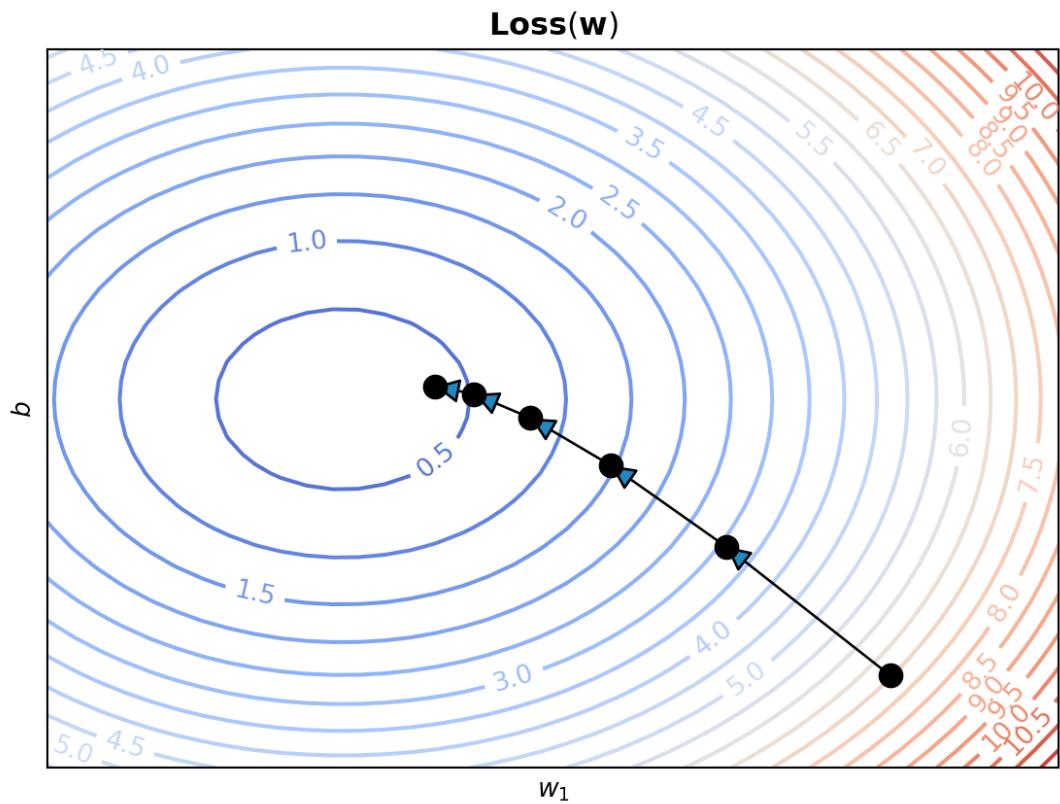
For i in 1,...,T : $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \nabla f(\mathbf{w}^{(i)})$



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

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

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



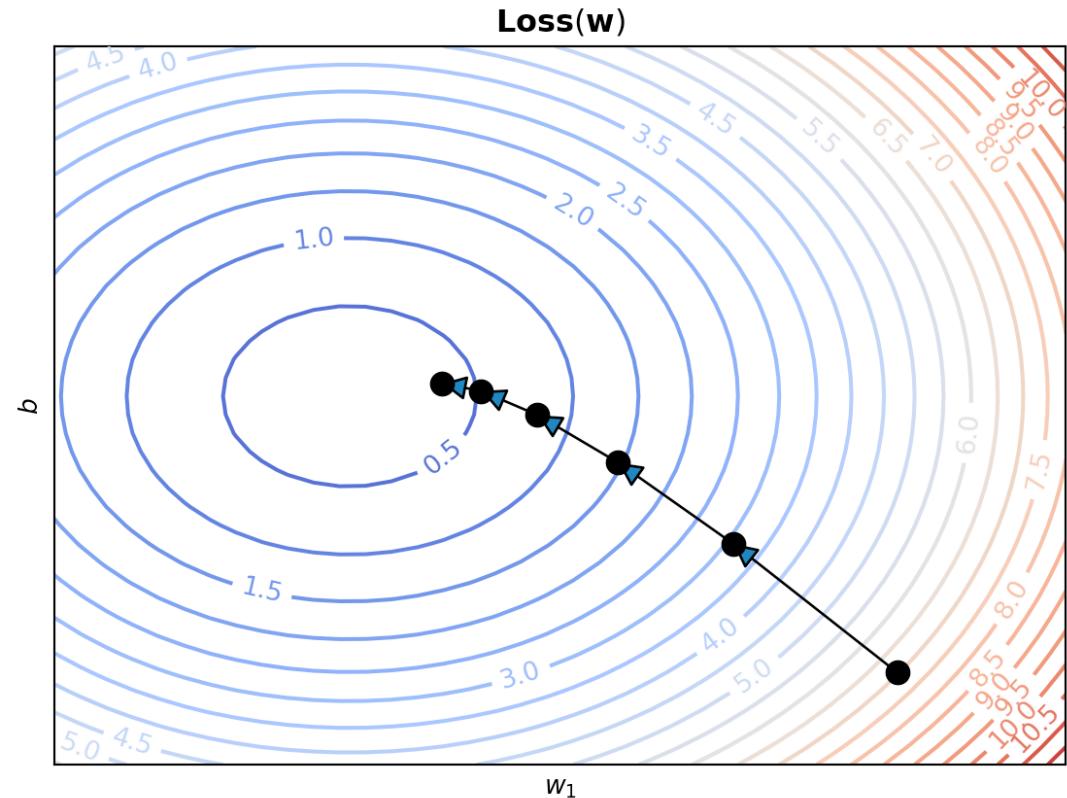
Recall that it's minimum value \mathbf{w}^* , a function f must have a gradient of $\mathbf{0}$.

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

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 $\mathbf{0}$.

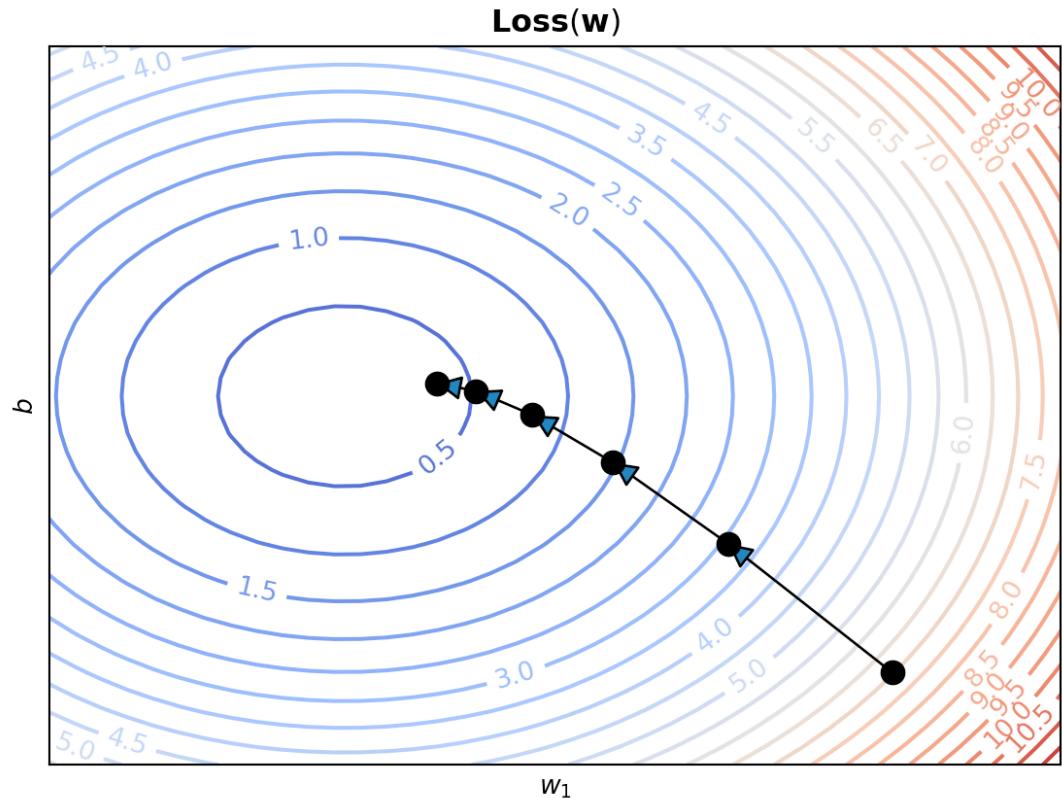
$$\nabla f(\mathbf{w}^*) = \mathbf{0}$$

It follows that:

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

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

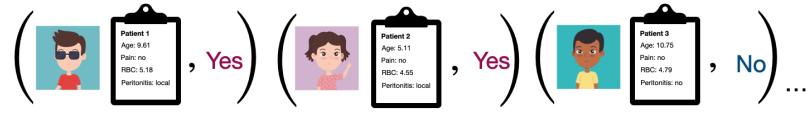
$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$$



Recap

Dataset

Set of known inputs and outputs



Input table

Output table

Patients with abdominal pain					
	Age	Appendix Size	Migratory Pain	RBC Count	RBC/Urine
0	8.61	0.0		4.18	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	10.51	9.0	no	4.82	low
6	10.83	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	10.42	4.5	yes	4.3	no
11	10.74	9.0	no	5.66	none
12	12.41	3.7	no	5.49	no
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	no
16	12.43	12.0	yes	4.62	none

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

Abstraction

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

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \dots & x_{Nn} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

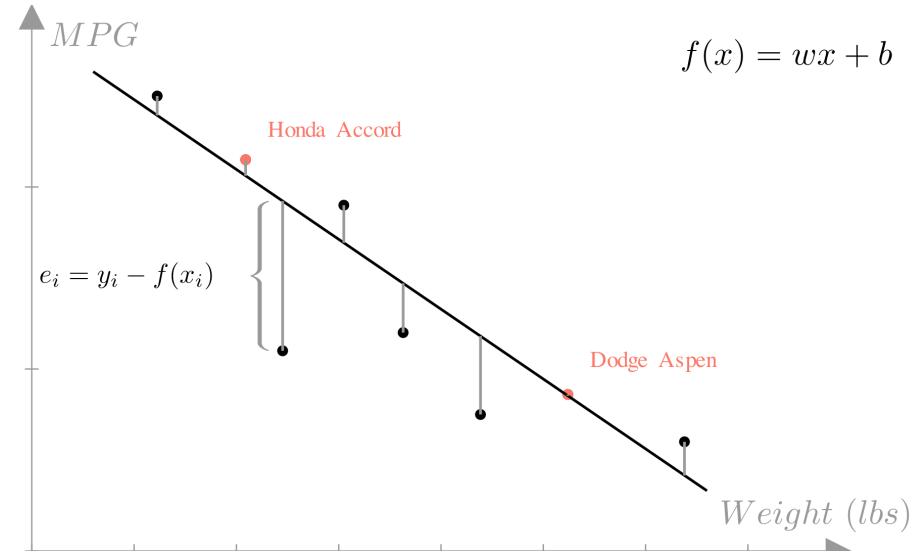
Linear (affine) prediction function

Input: \mathbf{x} _{*predict*} Output: y , $y = f(\mathbf{x})$

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

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)



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

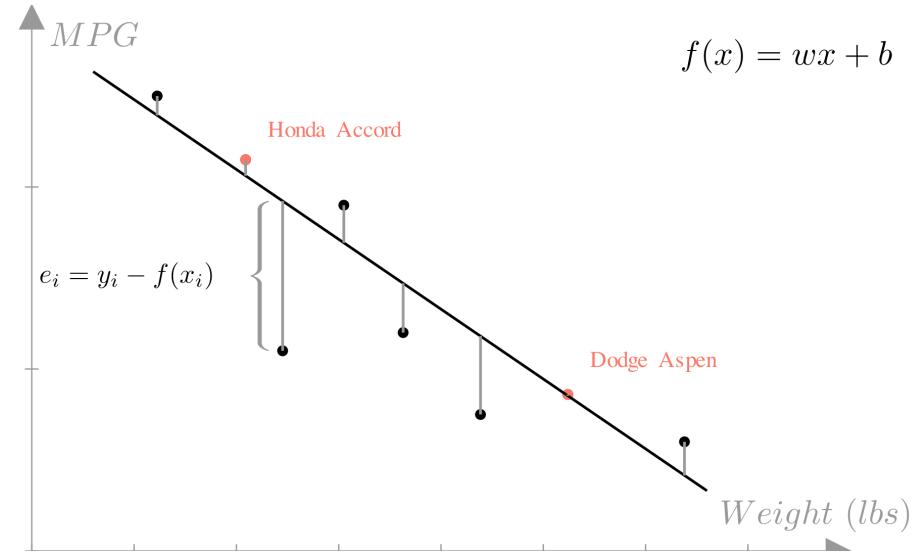
Linear (affine) prediction function

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

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

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

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



Augmenting inputs trick

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

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

We can easily see then that using this notation:

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

Linear (affine) prediction function

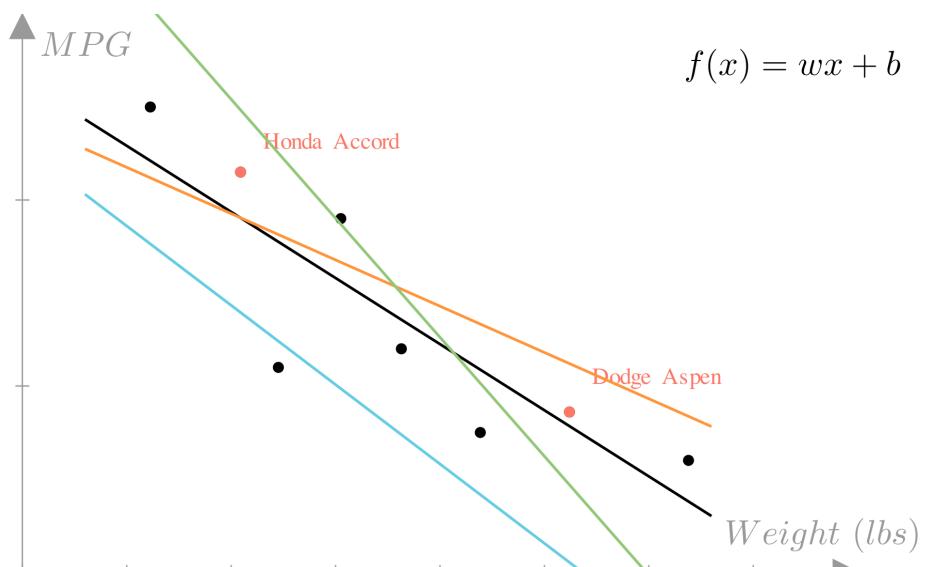
Input: $\mathbf{x}_{predict}$ Output: y , $y = f(\mathbf{x})$

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

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

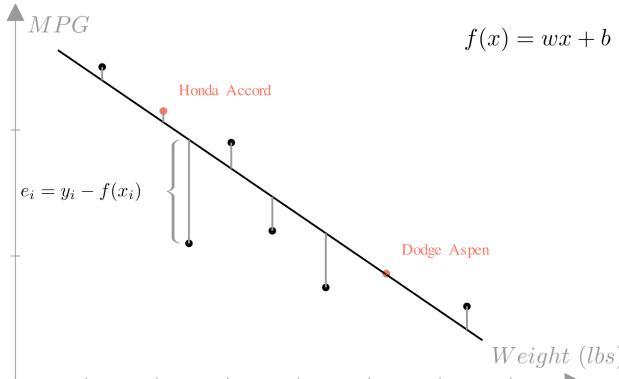
Affine function: $f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} + b$, Parameters: (Weights: \mathbf{w} , Bias: b)

How to choose a function?



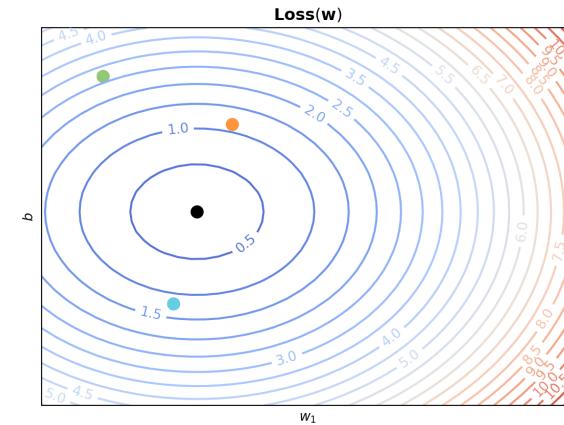
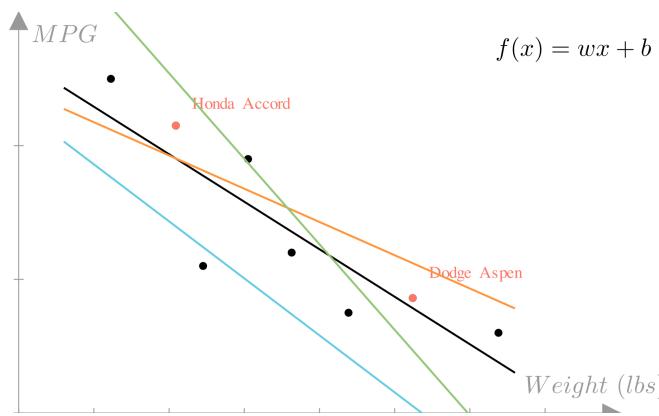
Residuals measure error

MSE loss minimizes squared residuals



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

Loss is a *function of parameters*



Find parameters that minimize loss:

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

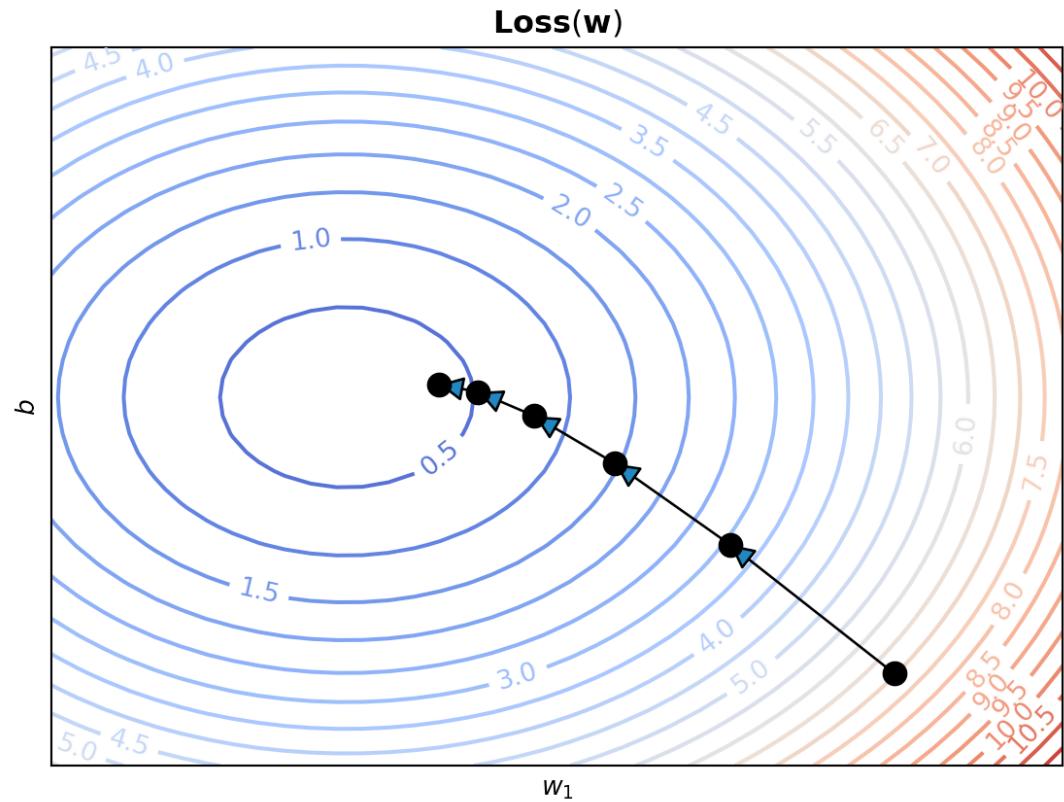
Gradient descent: start with random guess, then *update* in direction of steepest descent (gradient)

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

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

Gradient: vector of partial derivatives

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



Calculus with vectors is similar to calculus with scalars!

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

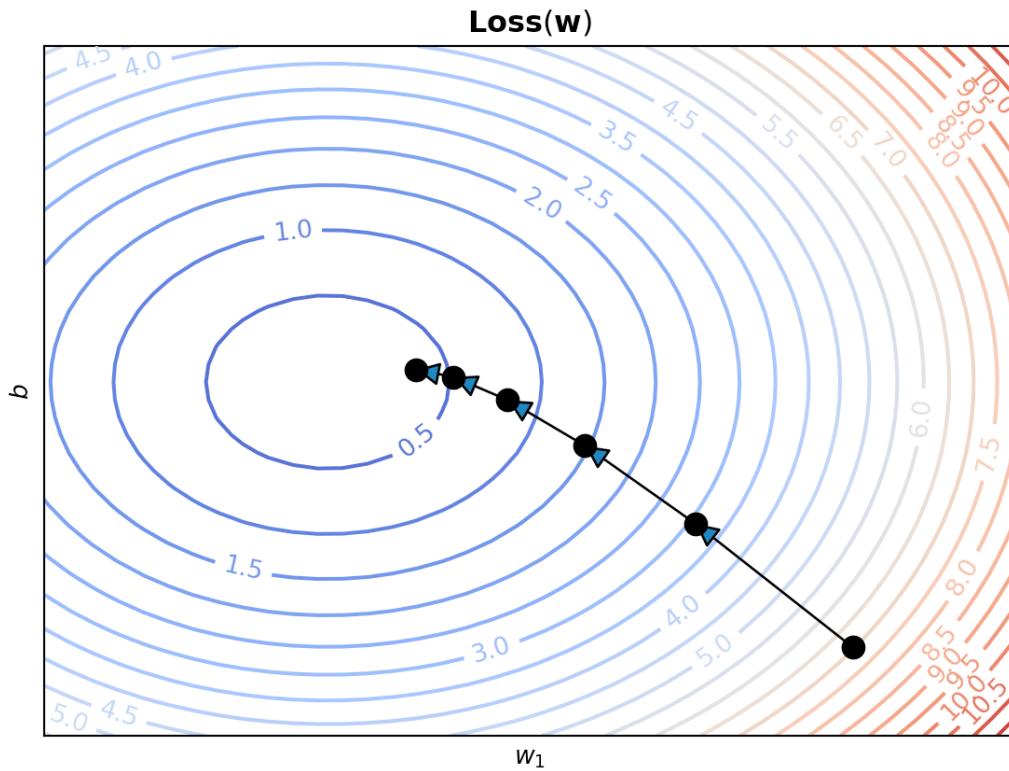
Can verify rules by taking individual partial derivatives

$$\frac{\partial}{\partial x_1} \mathbf{x}^T \mathbf{x} =$$

$$\frac{\partial}{\partial x_1} \sum_{i=1}^n x_i x_i = \frac{\partial}{\partial x_1} (x_1^2 + x_2^2 + x_3^2)$$

$$= \frac{\partial}{\partial x_1} x_1^2 = 2x_1$$

$$\frac{d}{d\mathbf{x}} \mathbf{x}^T \mathbf{x} = 2\mathbf{x}$$



Problems with gradient descent

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

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

It follows that:

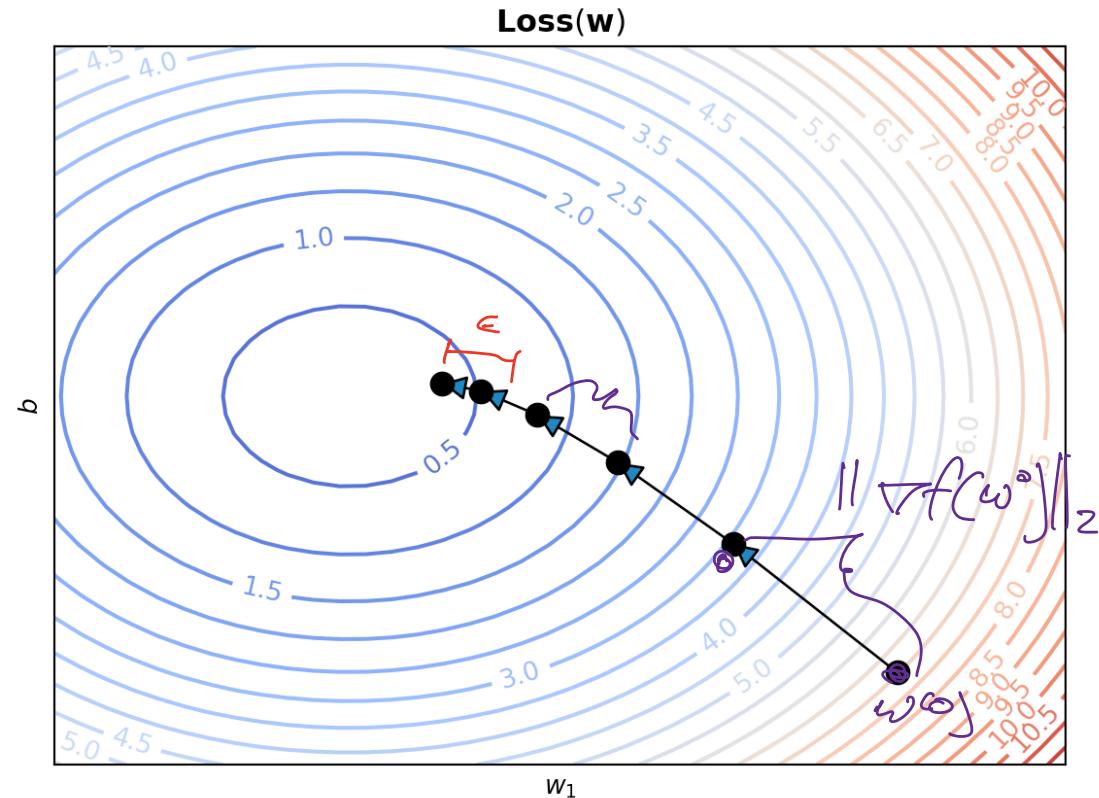
$$\mathbf{w}^* = \mathbf{w}^* - \nabla f(\mathbf{w}^*)$$

$$\nabla f$$

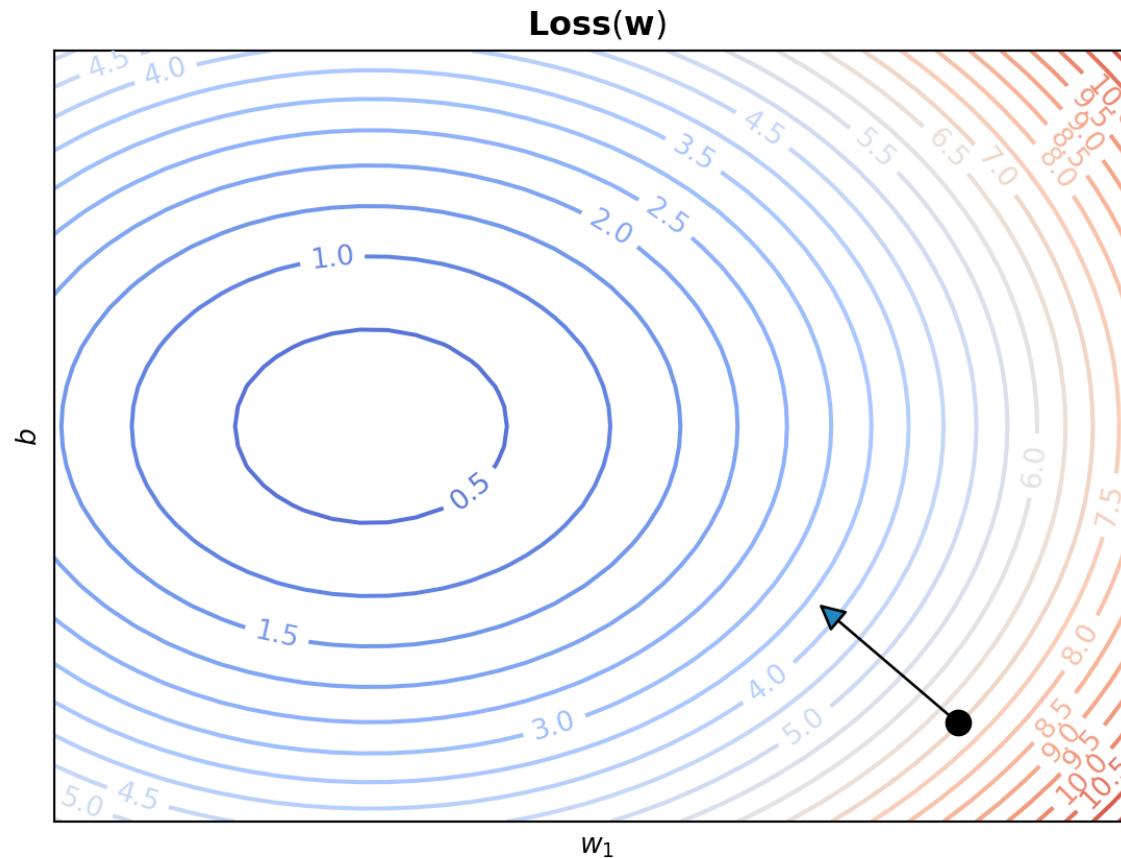
$$f \leftarrow f$$

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

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

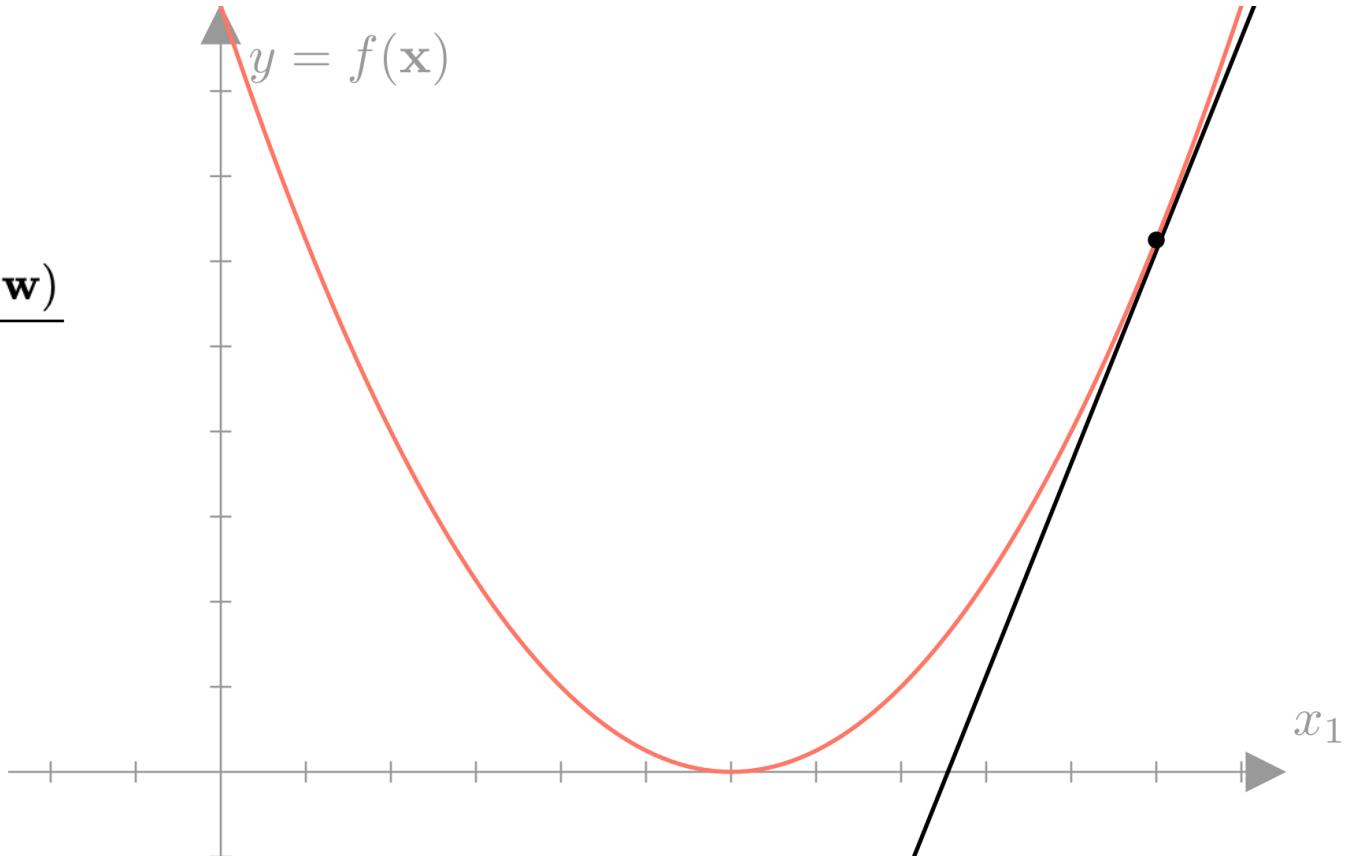


How big a step to take?

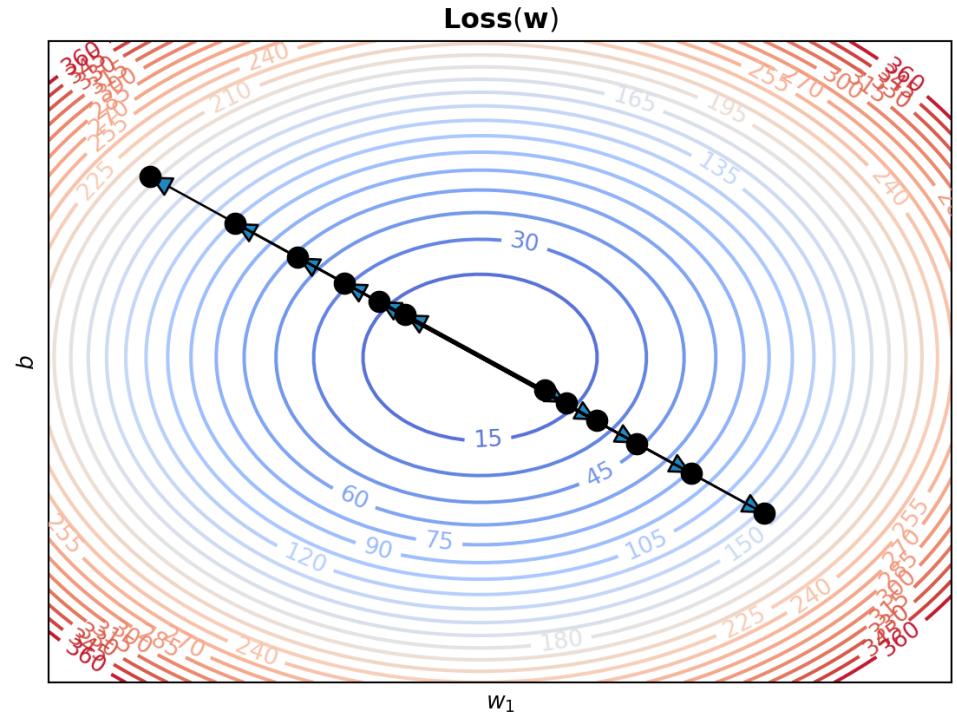
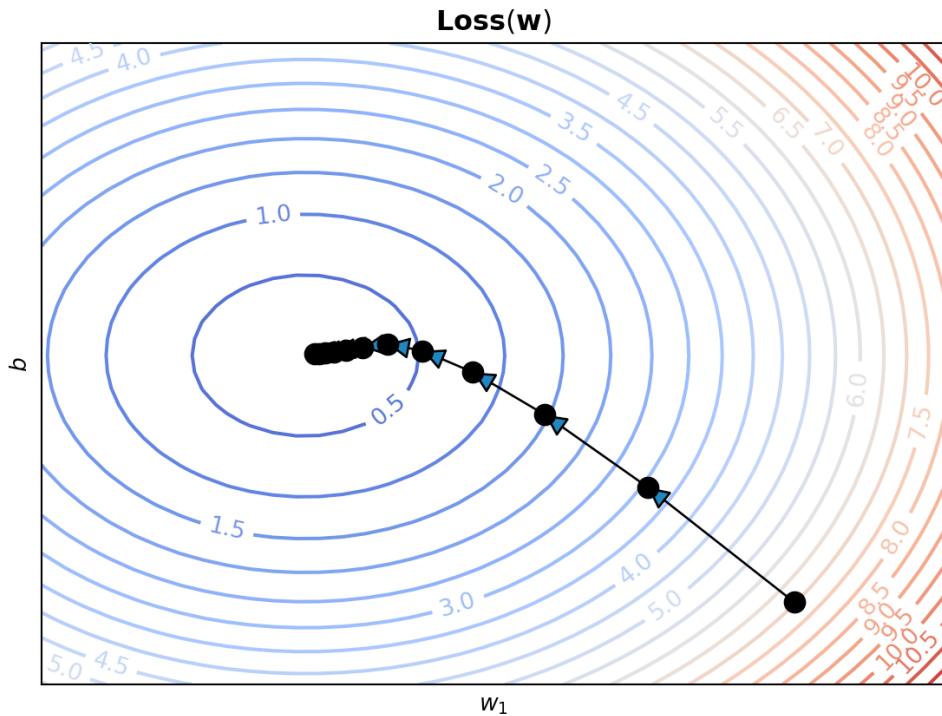


Gradient/derivative gives a *linear approximation of function*

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

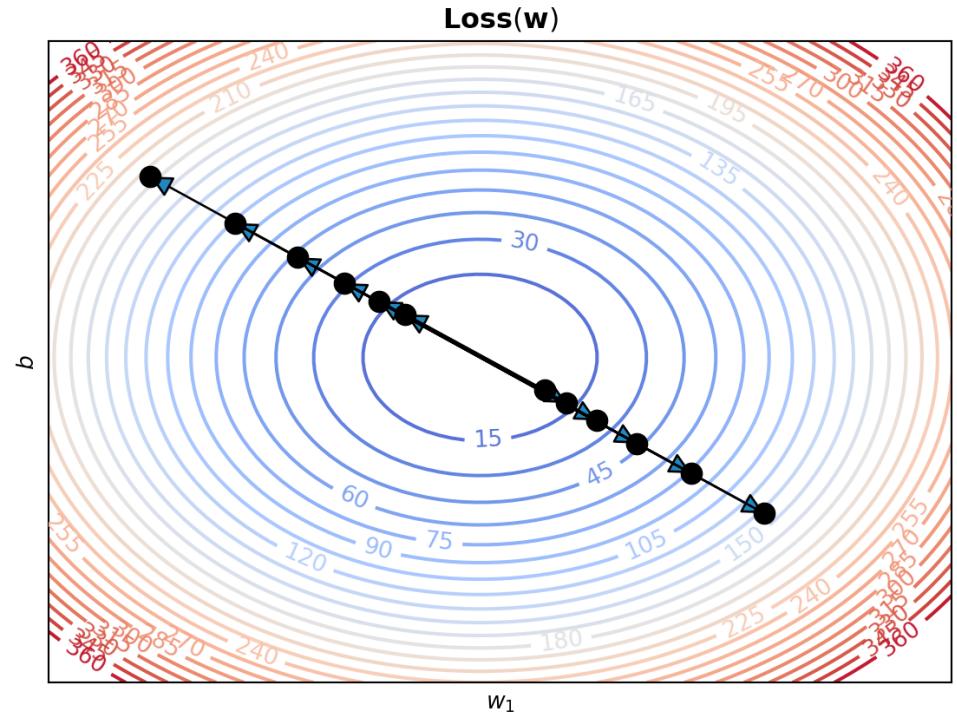
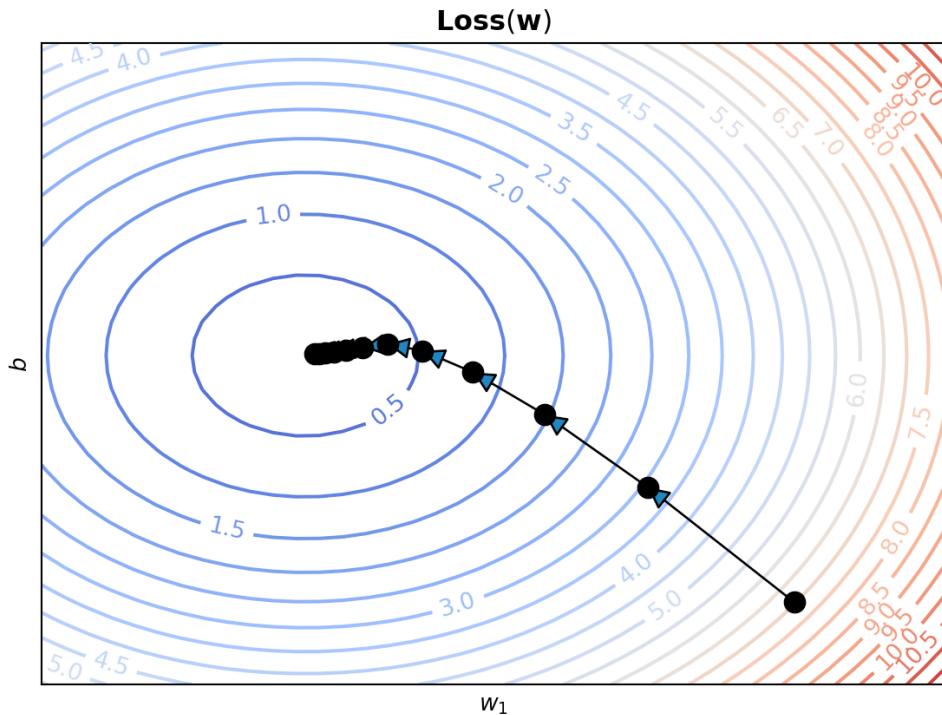


Gradients too big/too small



Step size!

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



Back to MSE

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

With this gradient our gradient descent update becomes:

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

Caveat

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

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

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

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

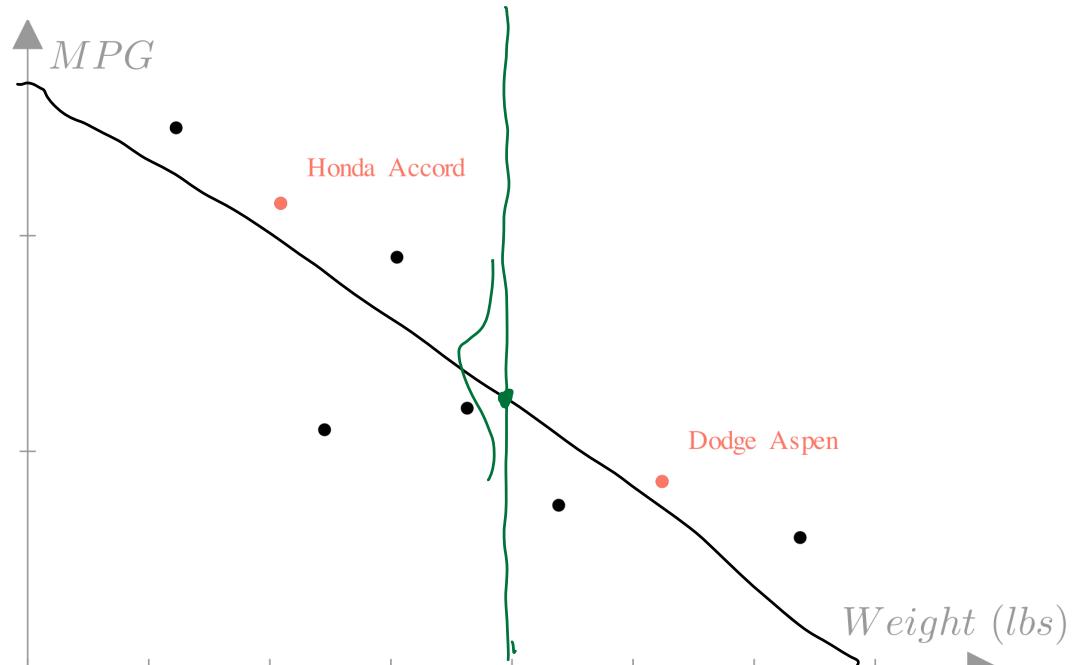
$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} (\mathbf{y} \mathbf{X})$$

Predictions are inexact

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

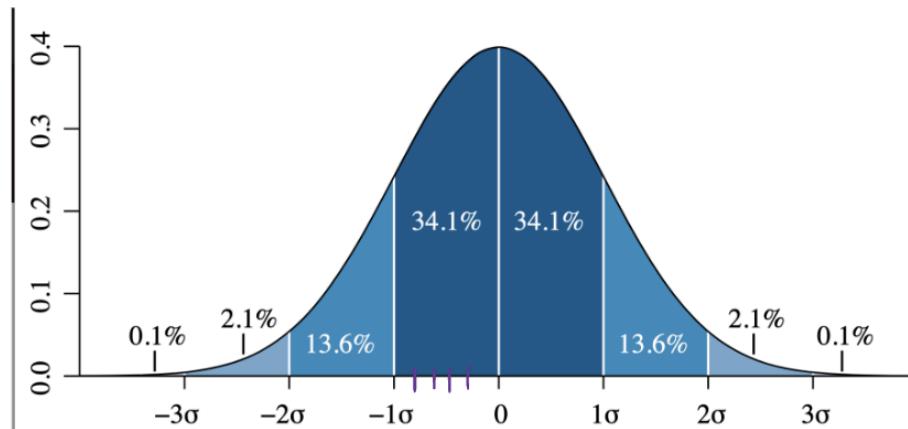
Predicted MPG = $f(\mathbf{x}) =$

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



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

Normal distribution



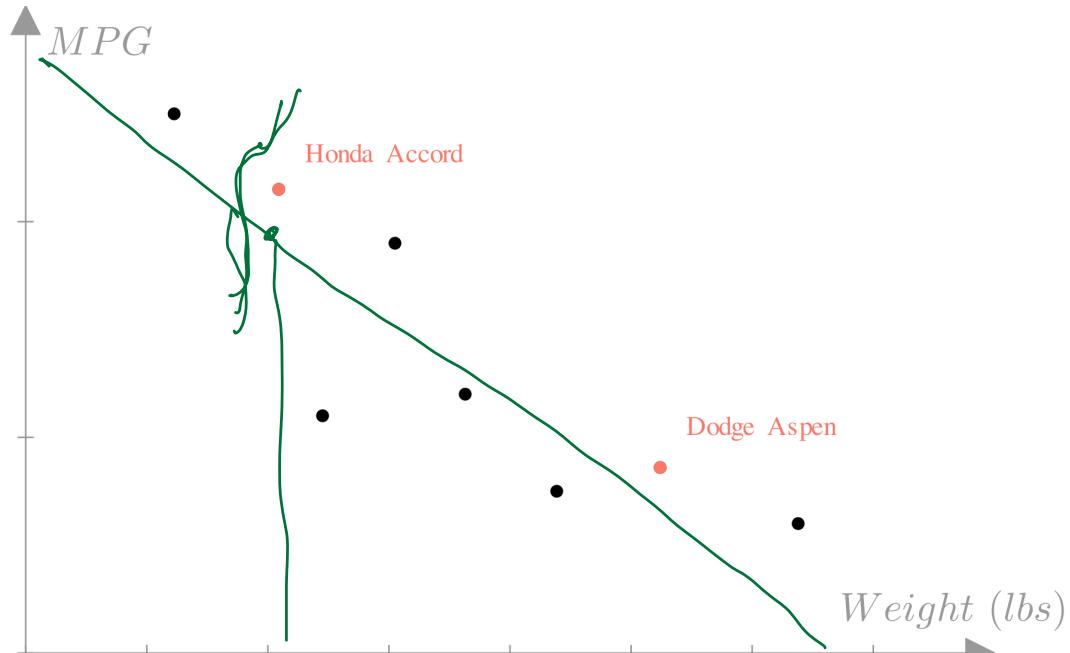
Normal distribution

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

Linear regression predicts mean!

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

↳ Linear Prediction



Normal distribution

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

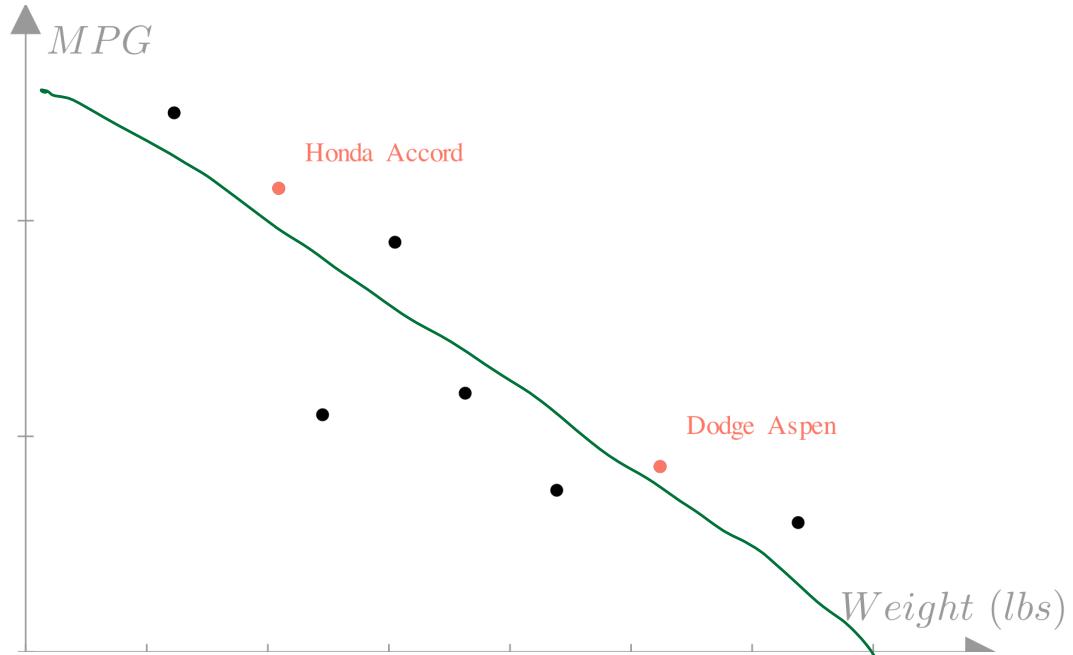
Linear regression predicts mean!

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

Conditional probability of observation

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

pdf of a normal dist.



Model labels

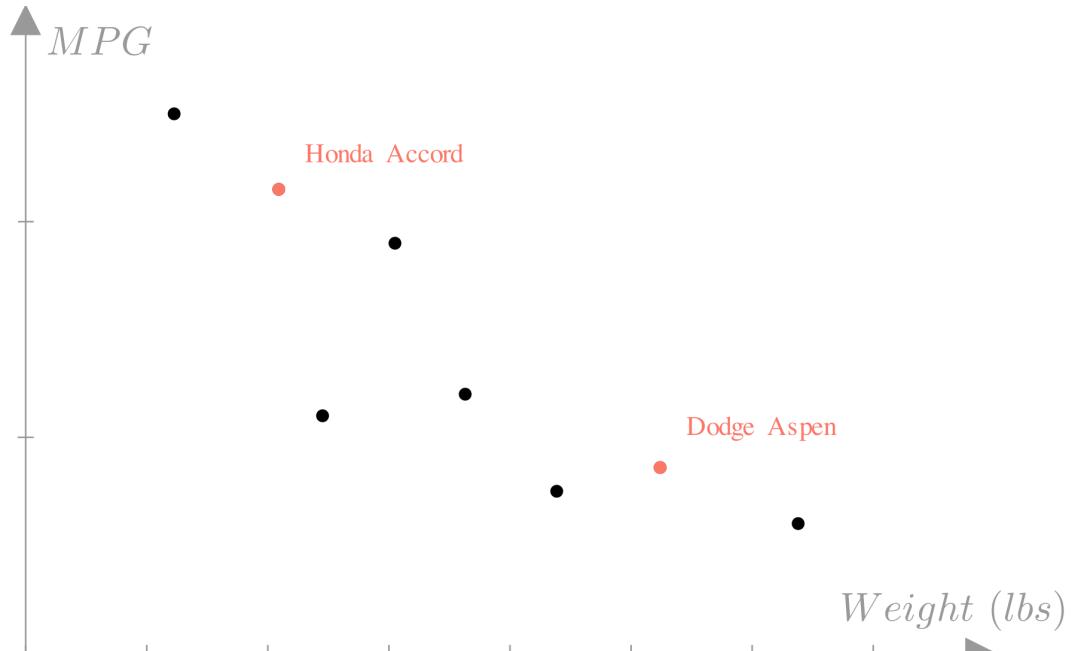
$$y_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma^2)$$

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

Model residuals

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

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



“Normal” linear regression model

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

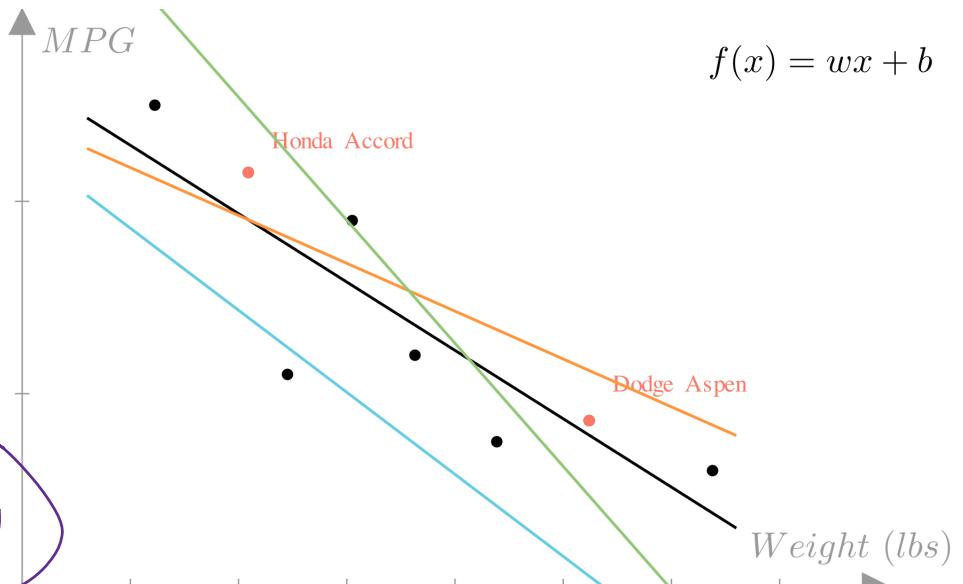
Likelihood of observation $y_i | x_i$

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

Loss function? Maximum likelihood

Max w

$$P(y_1, y_2, \dots | x_1, x_2, x_3, \dots, w)$$



“Normal” linear regression model

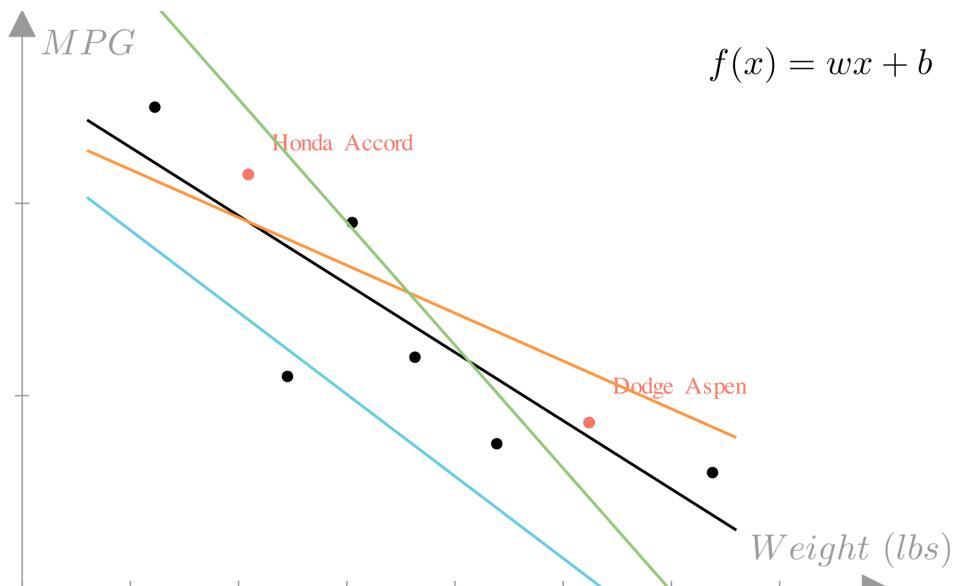
$$y_i \sim \mathcal{N}(\mathbf{x}_i^T \mathbf{w}, \sigma^2)$$

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

Loss function?

Goal: Maximize probability

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



$$\text{Independent} \Leftrightarrow P(\gamma_1, \gamma_2 | x_1, x_2, w) = P(\gamma_1 | x_1, w) \cdot P(\gamma_2 | x_2, w)$$

$$\rightarrow \arg \max_w \prod_{i=1}^N P(\gamma_i | x_i, w)$$

$$\rightarrow \arg \max_w \log \prod_{i=1}^N P(\gamma_i | x_i, w)$$

$$\rightarrow \arg \max_w \sum_{i=1}^N \log P(\gamma_i | x_i, w)$$

$$\rightarrow \arg \min_w - \sum_{i=1}^N \log P(\gamma_i | x_i, w)$$



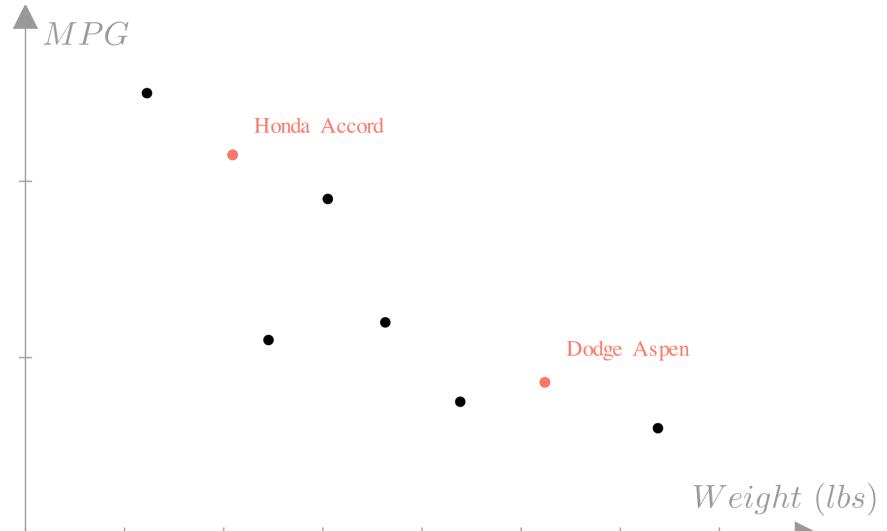
Negative Log-Likelihood

NLL

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

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

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



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

$$\log e^x = \log(x) \equiv \ln(x)$$

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

= - \sum_{i=1}^N \left[\log \frac{1}{\sigma\sqrt{2\pi}} + \left(-\frac{1}{2\sigma^2} (y_i - \mathbf{x}_i^T \mathbf{w})^2 \right) \right]

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

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

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

(Term 1)

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

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

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

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