Linear Regression

Univariate linear regression and gradient descent

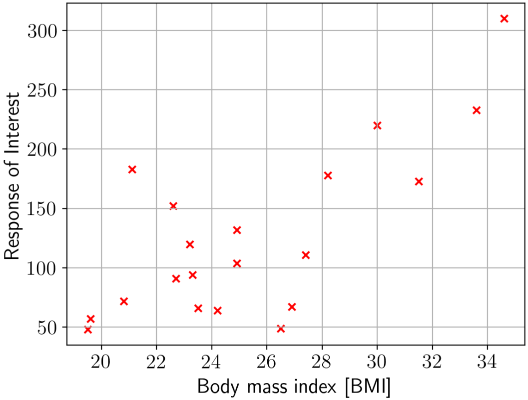
Khiem Nguyen

Email	khiem.nguyen@glasgow.ac.uk
MS Teams	khiem.nguyen@glasgow.ac.uk
Whatsapp	+44 7729 532071 (Emergency only)

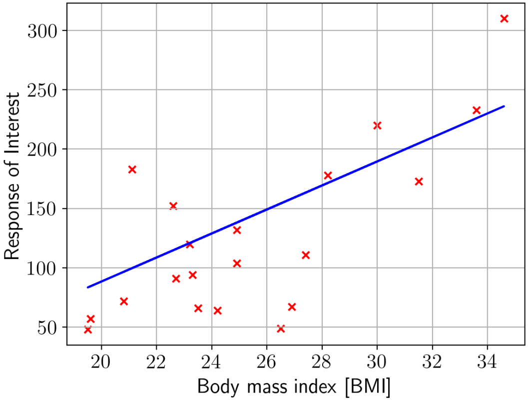
May 18, 2025

University of Glasgow

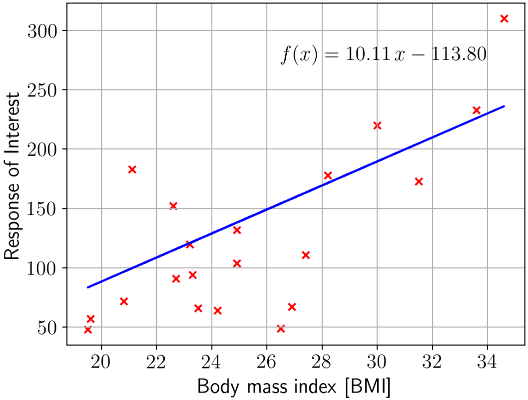
Linear regression: presentation



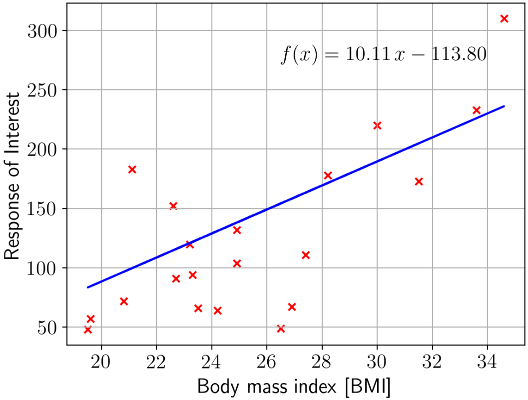
Linear regression: presentation



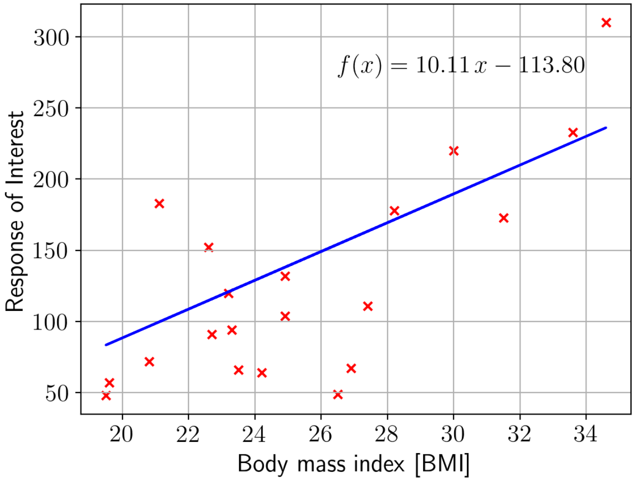
Linear regression: presentation



Linear regression: presentation



- Supervised learning model data has “right answers”.
- Regression model predicts numbers.
- Classification model predicts categories.



BMI	RI
32.1	151
21.6	75
30.5	141
25.3	206
...	...
23	135

Terminology

Training Data used to train the model

	x BMI	y RI
(1)	32.1	151
(2)	21.6	75
(3)	30.5	141
(4)	25.3	206
...	...	...
(m)	23	135

Terminology

Training Data used to train the model

	x	y
	BMI	RI
(1)	32.1	151
(2)	21.6	75
(3)	30.5	141
(4)	25.3	206
...	...	...
(m)	23	135

Notation

x = input variable
feature
y = output variable
target variable

Terminology

Training Data used to train the model

	x	y
	BMI	RI
(1)	32.1	151
(2)	21.6	75
(3)	30.5	141
(4)	25.3	206
...	...	...
(m)	23	135

Notation

x = input variable
feature

y = output variable
target variable

(x, y) = single training example

$(x^{(i)}, y^{(i)}) = i^{\text{th}}$ training example

We have $(1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}, \cdots, N^{\text{th}})$

Terminology

Training Data used to train the model

	x	y
	BMI	RI
(1)	32.1	151
(2)	21.6	75
(3)	30.5	141
(4)	25.3	206
...	...	...
(m)	23	135

$x^{(1)} = 32.1, \quad y^{(1)} = 151$

$x^{(2)} = 21.6, \quad y^{(2)} = 75$

$(x^{(1)}, y^{(1)}) = (32.1, 151)$

Note: $x^{(2)} \neq x^2 \rightarrow$ not exponent, just indexing

Notation

x = input variable
feature

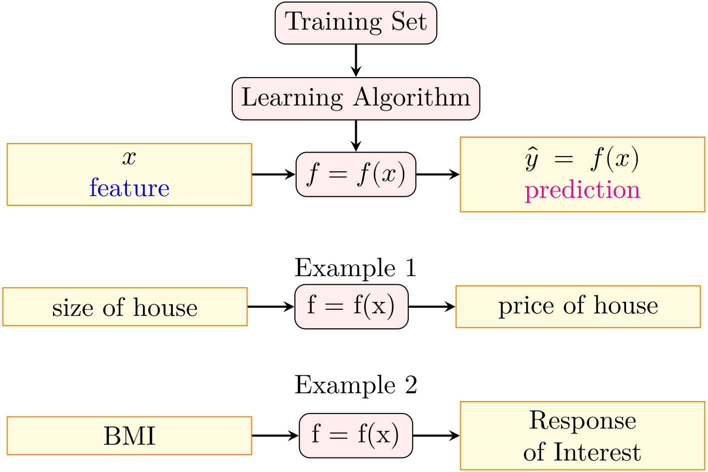
y = output variable
target variable

(x, y) = single training example

$(x^{(i)}, y^{(i)}) = i^{\text{th}}$ training example

We have $(1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}, \cdots, N^{\text{th}})$

Learning a hypothesis/function model

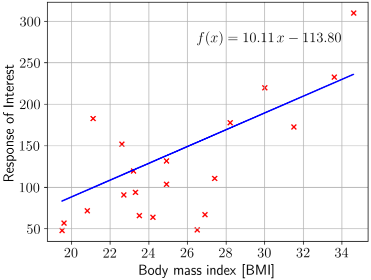


Linear function: $f(x) = wx + b, \quad w, b \in \mathbb{R}$

Learning a hypothesis/function model

$$x \text{ input} \longrightarrow \boxed{f = f(x)} \longrightarrow \hat{y} = f(x) \text{ predcition}$$

$$f(x) = f_{w,b}(x) = wx + b \tag{1}$$



- Linear regression with one variable (single feature x)
- $\underbrace{\text{Univariate}}_{\text{one variable}}$ linear regression

Interpretation of linear model

x	y
32.1	151
21.6	75
30.5	141
25.3	206
...	...
23	135

Interpretation of linear model

Model

$$f_{w,b}(x) = wx + b, \qquad w, b \quad - \quad \text{parameters}$$

x	y
32.1	151
21.6	75
30.5	141
25.3	206
...	...
23	135

Interpretation of linear model

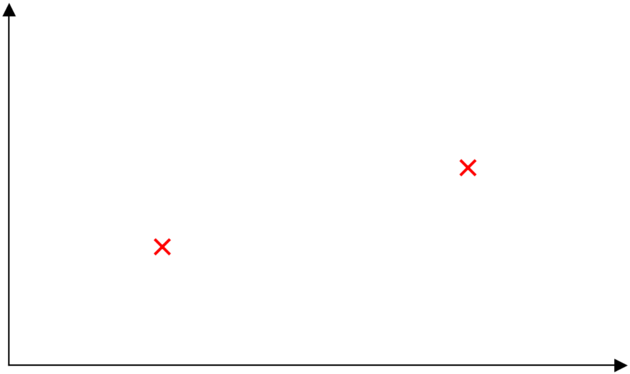
Model

$f_{w,b}(x) = wx + b, \qquad w, b \quad - \quad \text{parameters}$

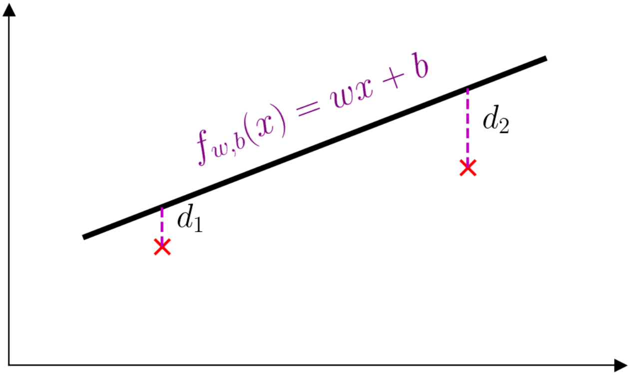
x	y
32.1	151
21.6	75
30.5	141
25.3	206
...	...
23	135

What do the beasts w, b do, and
Where (how) to find them?

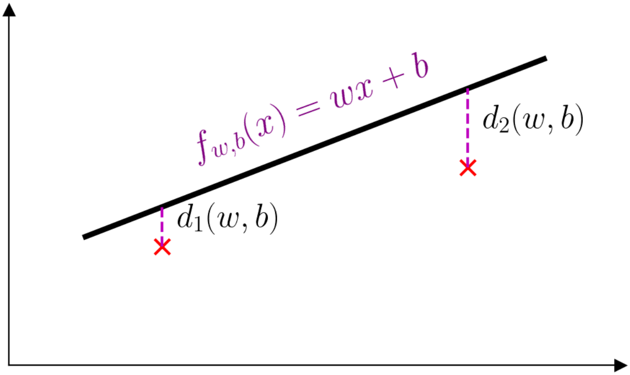
Cost function: detailed explanation



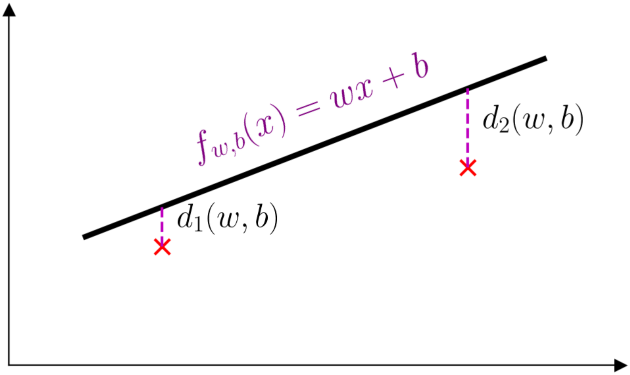
Cost function: detailed explanation



Cost function: detailed explanation



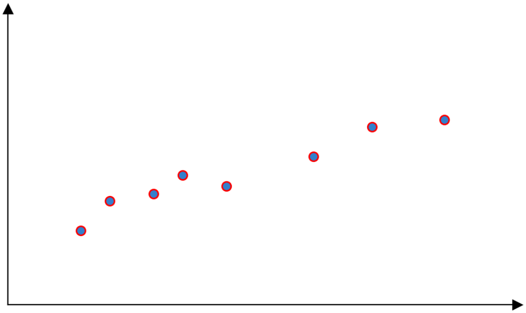
Cost function: detailed explanation



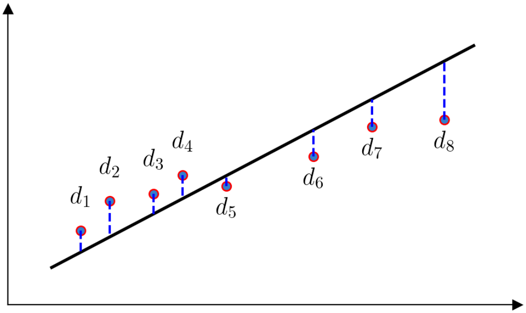
Minimize the total distance $[d_1(w,b) + d_2(w,b)]$ with respect to w, b

$$\min_{w,b}[d_1(w,b) + d_2(w,b)]$$

Cost function: detailed explanation

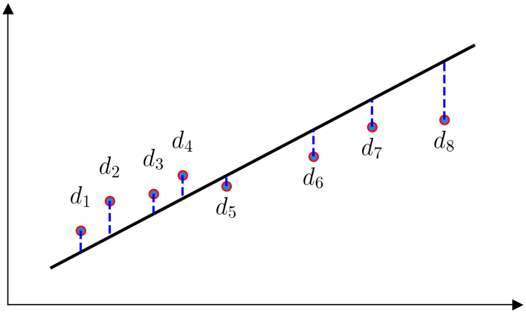


Cost function: detailed explanation



Many more data points $\quad\rightarrow\quad$ many more distances $d_j = d_j(w, b)$, $j = 1, \dots, m$

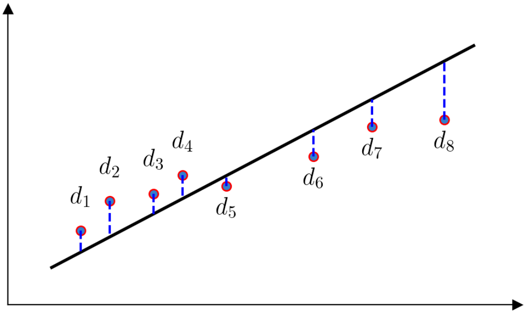
Cost function: detailed explanation



Many more data points $\rightarrow$ many more distances $d_j = d_j(w, b)$, $j = 1, \ldots, m$

$$\min_{w,b} \left\{ \left[d_1(w, b) + \cdots + d_m(w, b) \right] \right\}$$

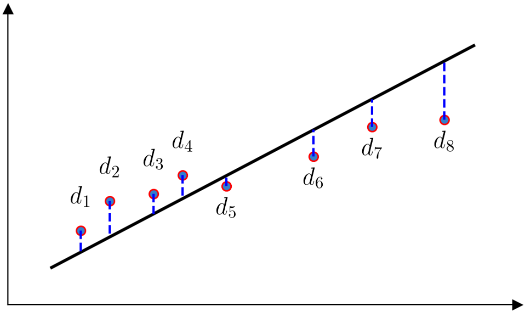
Cost function: detailed explanation



Many more data points $\rightarrow$ many more distances $d_j = d_j(w, b)$, $j = 1, \dots, m$

$$\min_{w,b} \left\{ \frac{1}{m} \Big[d_1(w, b) + \cdots + d_m(w, b) \Big] \right\}$$

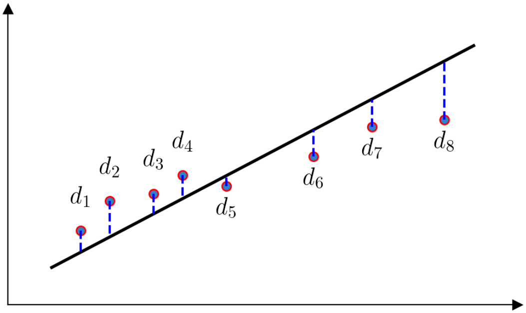
Cost function: detailed explanation



Many more data points $\rightarrow$ many more distances $d_j = d_j(w, b)$, $j = 1, \ldots, m$

$$\min_{w,b} \left\{ \frac{1}{m} \Big[d_1(w, b) + \cdots + d_m(w, b) \Big] \right\}$$

Cost function: detailed explanation



Many more data points $\quad\rightarrow\quad$ many more distances $d_j = d_j(w, b)$, $j = 1, \ldots, m$

$$\min_{w,b}\left\{\frac{1}{m}\sum_{i=1}^{m} d_i(w, b)]\right\}$$

Cost function: detailed explanation

But ... life is always more complicated than it looks ☺

Cost function: detailed explanation

But ... life is always more complicated than it looks ☺

Let us look at three data points $\{\mathbf{p}^{(1)} = (1,3),\ \mathbf{p}^{(2)} = (2,4),\ \mathbf{p}^{(3)} = (3,5)\}$:

$$\min_{w,b} G(w,b) = \min_{w,b} \frac{1}{3}\Big(|w \times 1 + b - 3| + |w \times 2 + b - 4| + |w \times 3 - 5|\Big)$$

Cost function: detailed explanation

But ... life is always more complicated than it looks ☺

Let us look at three data points $\{\mathbf{p}^{(1)} = (1,3),\ \mathbf{p}^{(2)} = (2,4),\ \mathbf{p}^{(3)} = (3,5)\}$:

$$\min_{w,b} G(w,b) = \min_{w,b} \frac{1}{3}\Big(|w \times 1 + b - 3| + |w \times 2 + b - 4| + |w \times 3 - 5|\Big)$$

- $G(w,b)$ is just linear in both w and b depending on the different regions
- We have to consider different regions to remove absolutes

Cost function: detailed explanation

But ... life is always more complicated than it looks ☺

Let us look at three data points $\{\mathbf{p}^{(1)} = (1,3),\ \mathbf{p}^{(2)} = (2,4),\ \mathbf{p}^{(3)} = (3,5)\}$:

$$\min_{w,b} G(w,b) = \min_{w,b} \frac{1}{3}\Big(|w \times 1 + b - 3| + |w \times 2 + b - 4| + |w \times 3 - 5|\Big)$$

- $G(w,b)$ is just linear in both w and b depending on the different regions
- We have to consider different regions to remove absolutes

But ... but how about this?

$$\min_{w,b} \mathcal{L}(w,b) = \frac{1}{3}\left[(w \cdot 1 + b - 3)^2 + (w \cdot 2 + b - 4)^2 + (w \cdot 3 + b - 5)^2\right] \tag{2}$$

- $\mathcal{L}(w,b)$ is a quadratic function in both w and b
- Finding the minimum of quadratic function is easy.

Cost function: Squared error cost function

$$\begin{aligned}\hat{y}^{(i)} &= f_{w,b}(x^{(i)}) \\ &= wx^{(i)} + b\end{aligned}$$

$$\begin{aligned}\mathcal{L}(w, b) &= \\ \\ &=\end{aligned}$$

Cost function: Squared error cost function

$$\begin{aligned}\hat{y}^{(i)} &= f_{w,b}(x^{(i)}) \\ &= wx^{(i)} + b\end{aligned}$$

$$\begin{aligned}\mathcal{L}(w,b) &= \left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &=\end{aligned}$$

Cost function: Squared error cost function

$$\begin{aligned}\hat{y}^{(i)} &= f_{w,b}(x^{(i)}) \\ &= wx^{(i)} + b\end{aligned}$$

$$\begin{aligned}\mathcal{L}(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &= \end{aligned}$$

Cost function: Squared error cost function

$$\begin{aligned}\hat{y}^{(i)} &= f_{w,b}(x^{(i)}) \\ &= wx^{(i)} + b\end{aligned}$$

$$\begin{aligned}\mathcal{L}(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &= \frac{1}{2m}\sum_{i}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2\end{aligned}$$

Cost function: Squared error cost function

$$\begin{aligned}\hat{y}^{(i)} &= f_{w,b}(x^{(i)}) \\ &= wx^{(i)} + b\end{aligned}$$

$$\begin{aligned}\mathcal{L}(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 \\ &= \frac{1}{2m}\sum_{i}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2\end{aligned}$$

Find w, b so that: $\hat{y}^{(i)}$ is close to $y^{(i)}$ for all data points $(x^{(i)}, y^{(i)})$.

$$\min_{w,b} \mathcal{L}(w,b)$$

Cost function: Intuition (again!)

- Model: $f_{w,b} = wx + b$
- Parameters: w, b
- Cost function:
$$\mathcal{L}(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{w,b}(x^{(i)}) - y^{(i)}\right)^2$$
- Minimization problem: $\min_{w,b\in\mathbb{R}} \mathcal{L}(w,b)$

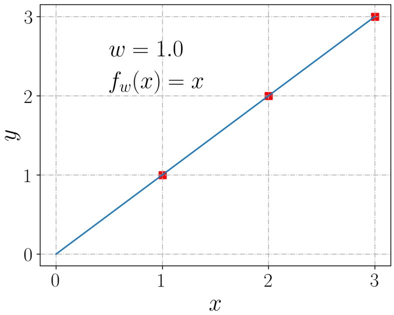
Simplified case: $b = 0$

$$f_{w,b=0}(x) := f_w(x) = wx$$

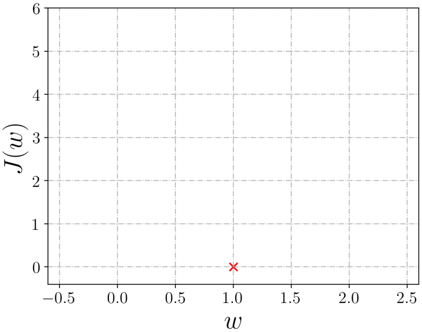
$$\mathcal{L}(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_w(x^{(i)} - y^{(i)}\right)^2$$

$$\longrightarrow \min_{w} \mathcal{L}(w)$$

Cost function: Intuition (again!)

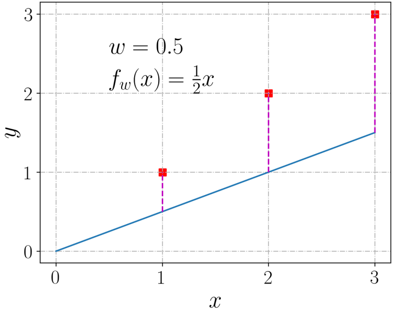


$$\mathcal{L}(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$$
$$= \frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})^2$$
$$= \frac{1}{2 \times 3}(0^2 + 0^2 + 0^2) = 0$$

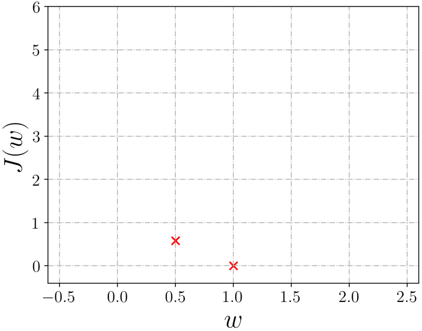


$$\mathcal{L}(w = 1, b = 0) = 0$$

Cost function: Intuition (again!)

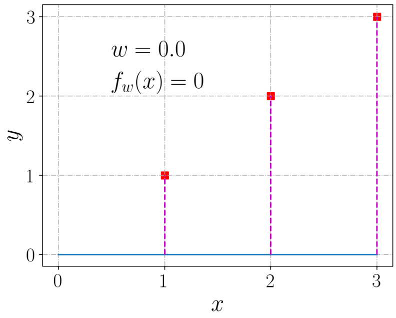


$$\mathcal{L}(w) = \frac{1}{2m}(wx^{(i)} - y^{(i)})^2$$
$$= \frac{1}{2 \times 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$
$$= 7/12$$

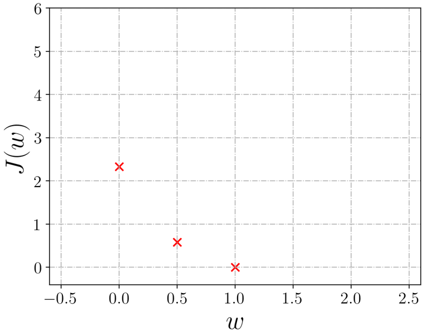


$$\mathcal{L}(w = 0.5, b = 0) = 0.58333$$

Cost function: Intuition (again!)

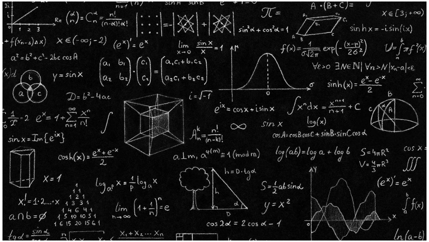


$$\mathcal{L}(w) = \frac{1}{2m}(wx^{(i)} - y^{(i)})^2$$
$$= \frac{1}{2 \times 3}\left[(0-1)^2 + (0-2)^2 + (0-3)^2\right]$$
$$= 14/6$$

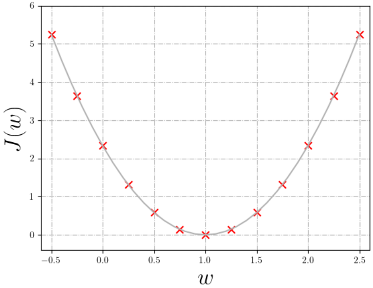


$$\mathcal{L}(w = 0, b = 0) = 14/3 \approx 2.3333$$

Cost function: Intuition (again!)



$$
\begin{aligned}
\mathcal{L}(w) &= \frac{1}{2 \times 3}\left[(w-1)^2 + (2w-2)^2 + (3w-3)^2\right] \\
&= \frac{1}{6}(1+4+9)(w-1)^2 \\
&= \frac{7}{3}(w-1)^2
\end{aligned}
$$



$$J(b=0) = \frac{7}{3}(w-1)^2$$

Visualize cost function

For our specific example with three data points $\{\mathbf{p}^{(1)} = (1,1), \mathbf{p}^{(2)} = (2,2), \mathbf{p}^{(3)} = (3,3)\}$:

$$\begin{aligned}
\mathcal{L}(w,b) &= \frac{1}{2\times 3}\left[(w\times 1 + b - 1)^2 + (w\times 2 + b - 2)^2 + (w\times 3 + b - 3)^2\right] \\
&= \frac{7}{3}(w-1)^2 + \frac{b^2}{2} + 2(w-1)b
\end{aligned}$$

For general cost function using linear regression model

$$\begin{aligned}
\mathcal{L}(w,b) &= \frac{1}{2m}\sum_{i=1}^{m}\Big(\color{red}{w}\,\underbrace{x^{(i)}}_{\in\mathbb{R}} + \color{red}{b} - \underbrace{y^{(i)}}_{\in\mathbb{R}}\Big)^2 \\
&= \sum_{i=1}^{m}(A_i w^2 + B_i b^2 + C_i wb + D_i), \qquad \text{where } A_i, B_i, C_i, D_i \text{ are all constants}
\end{aligned}$$

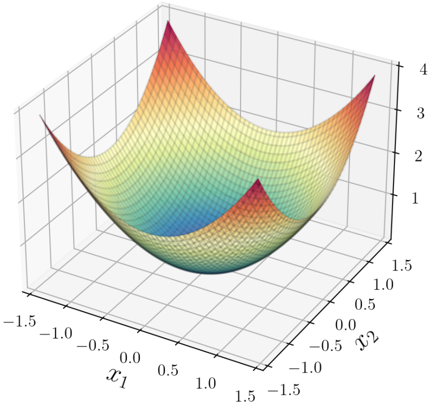
No matter how complicated the data would be, $\mathcal{L}(w,b)$ is a quadratic function of (w,b):

$$\mathcal{L}(w,b) = \gamma_{11}w^2 + \gamma_{12}wb + \gamma_{21}bw + \gamma_{22}b^2 + \gamma_{00},$$

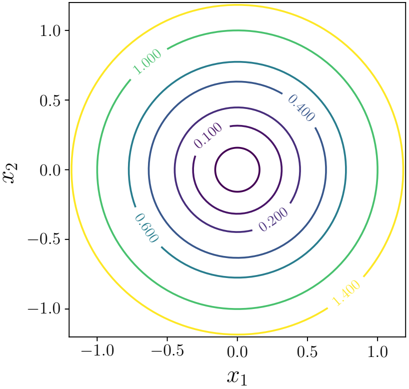
$$\gamma_{11}, \gamma_{12}, \gamma_{21}, \gamma_{22}, \gamma_{00} \in \mathbb{R}$$

Visualize cost function

$$f(x_1, x_2) = x_1^2 + x_2^2$$



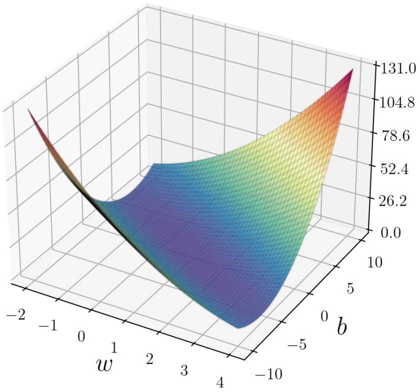
(a) Bowl shape quadratic function $x_1^2 + x_2^2$



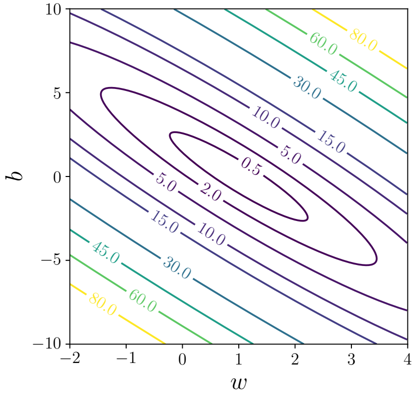
(b) Contour plot of quadratic function $x_1^2 + x_2^2$

Visualize cost function

$$\mathcal{L}(w,b) = \frac{7}{3}(w-1)^2 + \frac{b^2}{2} + 2(w-1)b$$



(a) Bowl shape cost function $\mathcal{L}(w,b)$



(b) Contour plot of $\mathcal{L}(w,b)$

Find minimization of a function

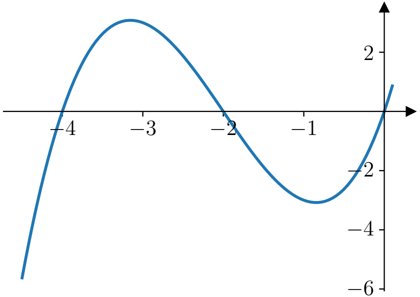
- Have some function $f(x_1, x_2, \ldots, x_n)$
- Want to minimize function f w.r.t. $(x_1, x_2, \ldots, x_n)$: $\min\limits_{x_1,\ldots,x_n} f(x_1, \ldots x_n)$
- The stationary point $\mathbf{x} = (x_1^*, \ldots, x_2^*)$ is the solution of

$$\nabla f(x_1, \ldots, x_n) = \mathbf{0} \quad \Longleftrightarrow \quad \begin{cases} \dfrac{\partial f}{x_1} = 0 \\ \cdots \\ \dfrac{\partial f}{x_n} = 0 \end{cases}$$

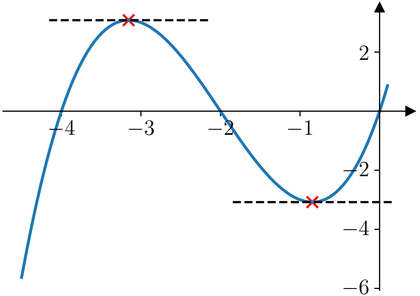
Last example: $\mathcal{L}(w, b) = \frac{7}{3}(w-1)^2 + \frac{b^2}{2} + 2(w-1)b$

$$\begin{cases} 0 = \dfrac{\partial J}{\partial w} = \dfrac{2}{3}(-7 + 3b + 7w) \\ 0 = \dfrac{\partial J}{\partial b} = -2 + b + 2w \end{cases} \quad \Rightarrow \quad \begin{cases} w = 1 \\ b = 0 \end{cases}$$

Find minimization of a function: Stationary point, 1D example

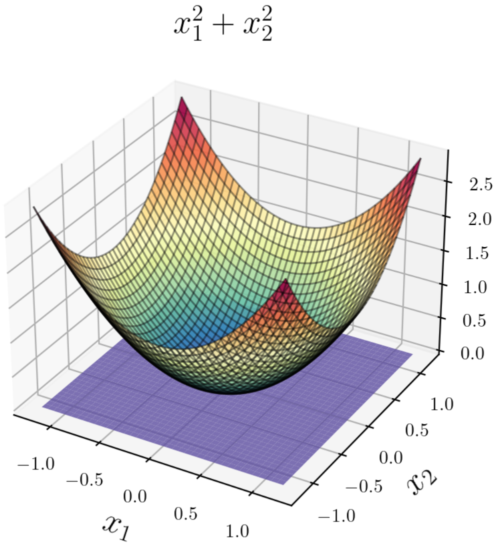


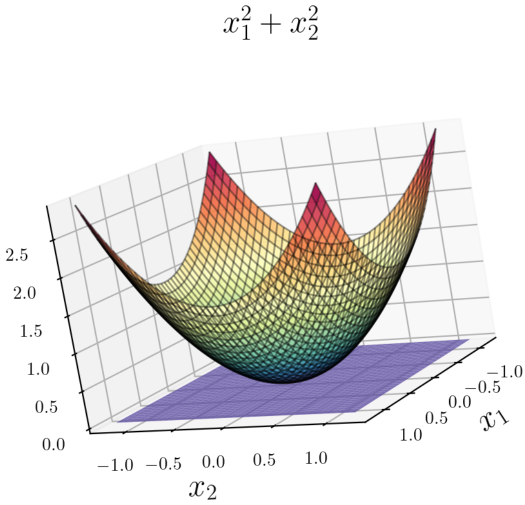
$$f(x) = x^3 + 6x^2 + 8x$$



$$f'(x) = 0 \rightarrow x = x_{\max}, x = x_{\min}$$

Find minimization of a function: Stationary point, 2D example





Gradient descent

Problem setting

- Have some function $f(\mathbf{x}) = f(x_1, \ldots, x_n)$
- Want to minimize function f w.r.t. $\mathbf{x} = (x_1, \ldots x_n)$

Simple idea

- Start with some initialization $(x_1^{[0]}, \ldots x_n^{[0]})$
- Update $\mathbf{x} = (x_1, \ldots, x_n)$ to reduce $f(x_1, \ldots, x_n)$
- Until we are already at or near a *minimum*

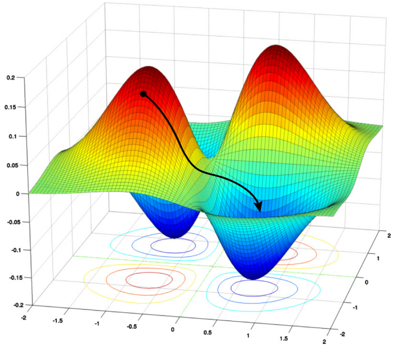
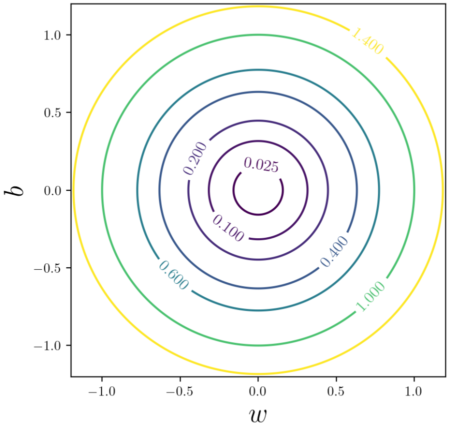


Figure: Gradient descent

Gradient descent: key idea – how to change function value at fastest rate?

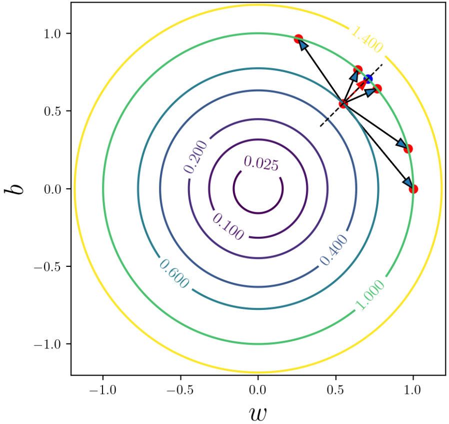
Stand at $(x_1^{[*]}, x_2^{[*]})$, what direction $\mathbf{n}$ ($|\mathbf{n}| = 1$) would give the fastest change in the value f?



Contour plot of $f(x_1, x_2) = x_1^2 + x_2^2$

Gradient descent: key idea – how to change function value at fastest rate?

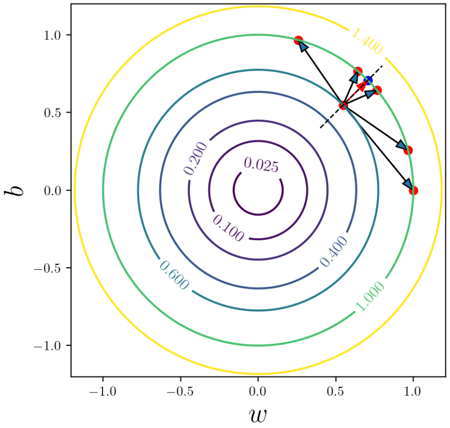
Stand at $(x_1^{[*]}, x_2^{[*]})$, what direction $\mathbf{n}$ ($|\mathbf{n}| = 1$) would give the fastest change in the value f?



Contour plot of $f(x_1, x_2) = x_1^2 + x_2^2$

Gradient descent: key idea – how to change function value at fastest rate?

Stand at $(x_1^{[*]}, x_2^{[*]})$, what direction $\mathbf{n}$ ($|\mathbf{n}| = 1$) would give the fastest change in the value f?

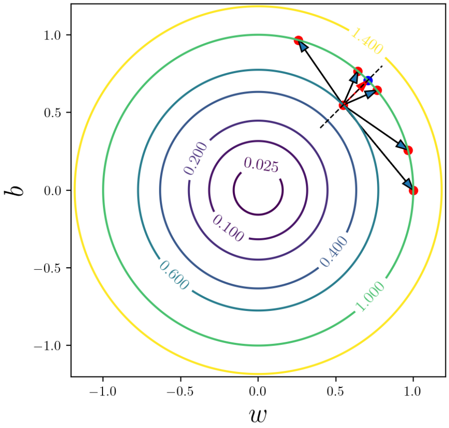


Contour plot of $f(x_1, x_2) = x_1^2 + x_2^2$

- f at $(x_1^{[*]}, x_2^{[*]})$ changes fastest in the direction, called here $\mathbf{n}$ ($|\mathbf{n}| = 1$), orthogonal to the contour line $\mathcal{C}$ going through $(x_1^{[*]}, x_2^{[*]})$
- The gradient of f at $(x_1^{[*]}, x_2^{[*]})$ is also orthogonal to the contour line $\mathcal{C}$.[a]

Gradient descent: key idea – how to change function value at fastest rate?

Stand at $(x_1^{[*]}, x_2^{[*]})$, what direction $\mathbf{n}$ ($|\mathbf{n}| = 1$) would give the fastest change in the value f?



Contour plot of $f(x_1, x_2) = x_1^2 + x_2^2$

- f at $(x_1^{[*]}, x_2^{[*]})$ changes fastest in the direction, called here $\mathbf{n}$ ($|\mathbf{n}| = 1$), orthogonal to the contour line $\mathcal{C}$ going through $(x_1^{[*]}, x_2^{[*]})$
- The gradient of f at $(x_1^{[*]}, x_2^{[*]})$ is also orthogonal to the contour line $\mathcal{C}$.[a]

➥ f changes fastest in the direction of the gradient of f:

$$\mathbf{n} = \frac{\nabla f(x_1^{[*]}, x_2^{[*]})}{|\nabla f(x_1^{[*]}, x_2^{[*]})|} = \frac{1}{|\nabla f|}\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)$$

[a] Please look at the supplementary note if you are interested.

Gradient descent algorithm

Repeat until convergence

$$\begin{cases} x_1^{[t+1]} = x_1^{[t]} - \alpha^{[t]} \dfrac{\partial}{\partial x_1} f(x_1^{[t]}, \dots, x_n^{[t]}) \\ \quad \dots \\ x_n^{[t+1]} = x_n^{[t]} - \alpha^{[t]} \dfrac{\partial}{\partial x_n} f(x_1^{[t]}, \dots, x_n^{[t]}) \end{cases} \Leftrightarrow \quad \mathbf{x}^{[t]} = \mathbf{x}^{[t]} - \alpha^{[t]} \nabla f(\mathbf{x}^{[t]}), \quad t = 0, \dots, \infty \tag{3}$$

$\frac{\partial}{\partial x_1} f(x_1, \dots, x_n)$	(partial) derivatives
$\alpha^{[t]}$	learning rate

- We must *simultaneously update* $\mathbf{x} = (x_1, \dots, x_n)$.
- We can keep or change $\alpha^{[t]}$ at iteration step.

Gradient descent algorithm

Correct: Simultaneous update

$$\text{tmp}x_1 = x_1 - \alpha \frac{\partial f}{\partial x_1}(x_1, x_2)$$
$$\text{tmp}x_2 = x_2 - \alpha \frac{\partial f}{\partial x_2}(x_1, x_2)$$
$$x_1 = \text{tmp}x_1$$
$$x_2 = \text{tmp}x_2$$

Incorrect: Update before compute deri.

$$\text{tmp}x_1 = x_1 - \alpha \frac{\partial}{x_1}$$
$$x_1 = \text{tmp}x_1$$
$$\text{tmp}x_2 = x_2 - \alpha \frac{\partial f}{\partial x_2}(\underbrace{x_1}_{\text{tmp}x_1}, x_2)$$

- Python code: *Correct*

```
df_dx1 = compute_df_dx1(x1, x2)
df_dx2 = compute_df_dx2(x1, x2)
tmpx1 = x1 - df_dx1
tmpx2 = x2 - df_dx2
x1 = tmpx1
x2 = tmpx2
```

- Python code: *Incorrect*

```
df_dx1 = compute_df_dx1(x1, x2)
tmpx1 = x1 - df_dx1
x1 = tmpx1  # don't do this!
# don't do this, x1 holds values tmpx1 now
df_dx2 = compute_df_dx2(x1, x2)
tmpx2 = x2 - df_dx2
x2 = tmpx2
```

Gradient descent algorithm

Coming back to our cost function $J = \mathcal{L}(w, b)$: $f \to J,\ x_1 \to w,\ x_2 \to b$

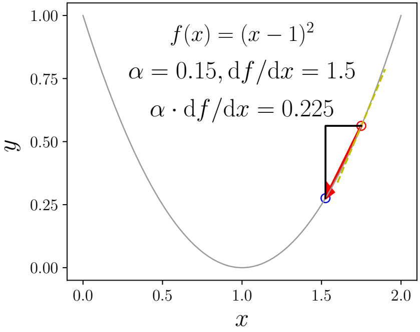
Repeat until convergence:

$$\begin{cases} w = w - \alpha \dfrac{\partial J}{\partial w}(w, b) \\ b = b - \alpha \dfrac{\partial J}{\partial b}(w, b) \end{cases} \tag{4}$$

- **Python code**: This is not efficient, just for illustration

```
dJ_dw = compute_dJ_dw(w, b)
dJ_db = compute_dJ_db(w, b)
tmp_w = w - dJ_dw
tmp_b = w - dJ_db
w = tmp_w
b = tmp_b
```

Gradient descent: why minus sign in front of α

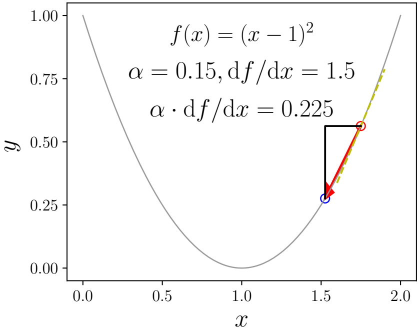


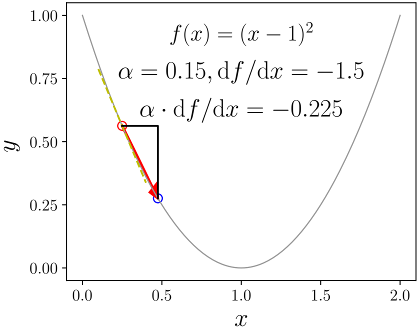
If $\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) > 0$, f is $\nearrow$ in a neighbor $(x_0 - \delta, x_0 + \delta)$.

To go in the opposite direction of $f \nearrow$, $x_0 \searrow$

$$x_{\text{next}} = x_0 - \underbrace{\alpha}_{>0} \cdot \underbrace{\mathrm{d}f/\mathrm{d}x}_{>0} < x_0$$

Gradient descent: why minus sign in front of α





If $\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) > 0$, f is $\nearrow$ in a neighbor $(x_0 - \delta, x_0 + \delta)$.

To go in the opposite direction of $f \nearrow$, $x_0 \searrow$

$$x_{\text{next}} = x_0 - \underbrace{\alpha}_{>0} \cdot \underbrace{\mathrm{d}f/\mathrm{d}x}_{>0} < x_0$$

If $\frac{\mathrm{d}f}{\mathrm{d}x}(x_0) < 0$, f is $\searrow$ in a neighbor $(x_0 - \delta, x_0 + \delta)$.

To go in the opposite direction of $f \searrow$, $x_0 \nearrow$

$$x_{\text{next}} = x_0 - \underbrace{\alpha}_{>0} \cdot \underbrace{\mathrm{d}f/\mathrm{d}x}_{<0} > x_0$$

Gradient descent: extra thought

How about we use the plus sign in front of α?

Where would it lead to?

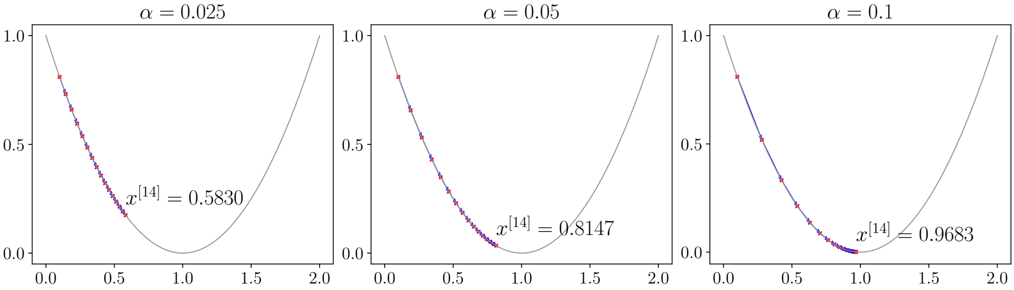
Gradient descent: extra thought

That's true!

It is the algorithm for solving the maximization problem.

Learning rate α

Let us consider $x = x - \alpha \frac{\mathrm{d}f}{\mathrm{d}x}$ with $f(x) = (x-1)^2$ and $x^{[0]} = 0.1$.

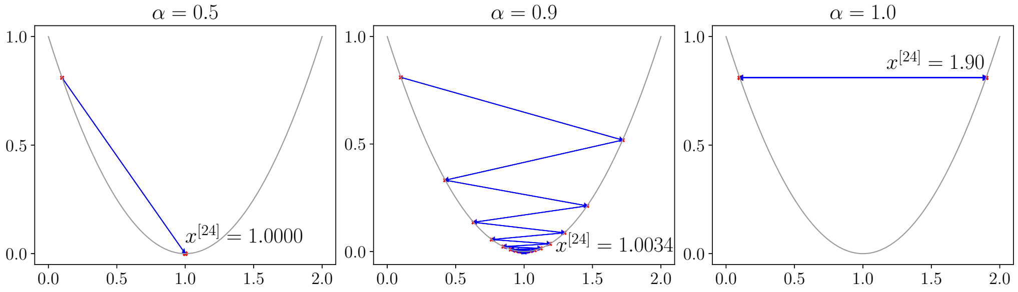


$\alpha = 0.025, x^{[14]} = 0.5830$ $\alpha = 0.05, x^{[14]} = 0.8147$ $\alpha = 0.1, x^{[14]} = 0.9683$

If α is too small, *gradient descent may be slow*.

Learning rate α

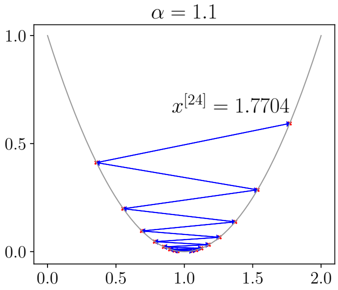
Let us consider $x = x - \alpha \frac{\mathrm{d}f}{\mathrm{d}x}$ with $f(x) = (x-1)^2$ and $x^{[0]} = 0.1$. We now increase α as follows



$\alpha = 0.5, x^{[24]} = 0.5830$ $\alpha = 0.9, x^{[24]} = 0.8147$ $\alpha = 1.0, x^{[24]} = 0.9683$

Learning rate α

- Let us initialize $x^{[0]} = 0.95$, much close to the solution $x_{\min} = 1$ but then use $\alpha = 1.1$



- If α is too large, gradient descent may
  - overshoot, never reach minimum
  - fail to converge, diverge
  - or be to slow too (see last slide – it just shoots over back and forth)

Batch gradient descent

Batch: Each step of gradient descent uses all the training examples.

	x	y
	BMI	RI
(1)	32.1	151
(2)	21.6	75
(3)	30.5	141
(4)	25.3	206
$\vdots$	$\vdots$	$\vdots$
(m)	23	135

Batch gradient descent uses all the training examples so that we calculate the derivatives of $\mathcal{L}(w,b)$ w.r.t. w and b by summing up over all training examples.

$$\partial J/\partial w = \sum_{\text{all training samples}} \cdots, \quad \partial J/\partial b = \sum_{\text{all training samples}} \cdots$$

'Other' gradient descent: Each step of gradient descent uses just subsets of the training examples. So, we use "approximate" partial derivatives of $\mathcal{L}(w,b)$ by summing up over a portion of training examples. We call this **minibatch gradient descent**.

$$\partial J/\partial w = \sum_{\text{subset of samples}} \cdots, \quad \partial J/\partial b = \sum_{\text{subset of samples}} \cdots$$

Training set versus test set

- **Training set**:

  Set of examples used for fitting/training the regression models

- **Test set**:

  Set of examples used for assessing how well the regression models perform or generalize to the unseen data (test data)

➥ We will learn in the next lectures some metrics to evaluate how well a regression model performs on a given data set:

- R squared score/coefficient of determination
- Mean squared error
- Accuracy score (for classification problem)