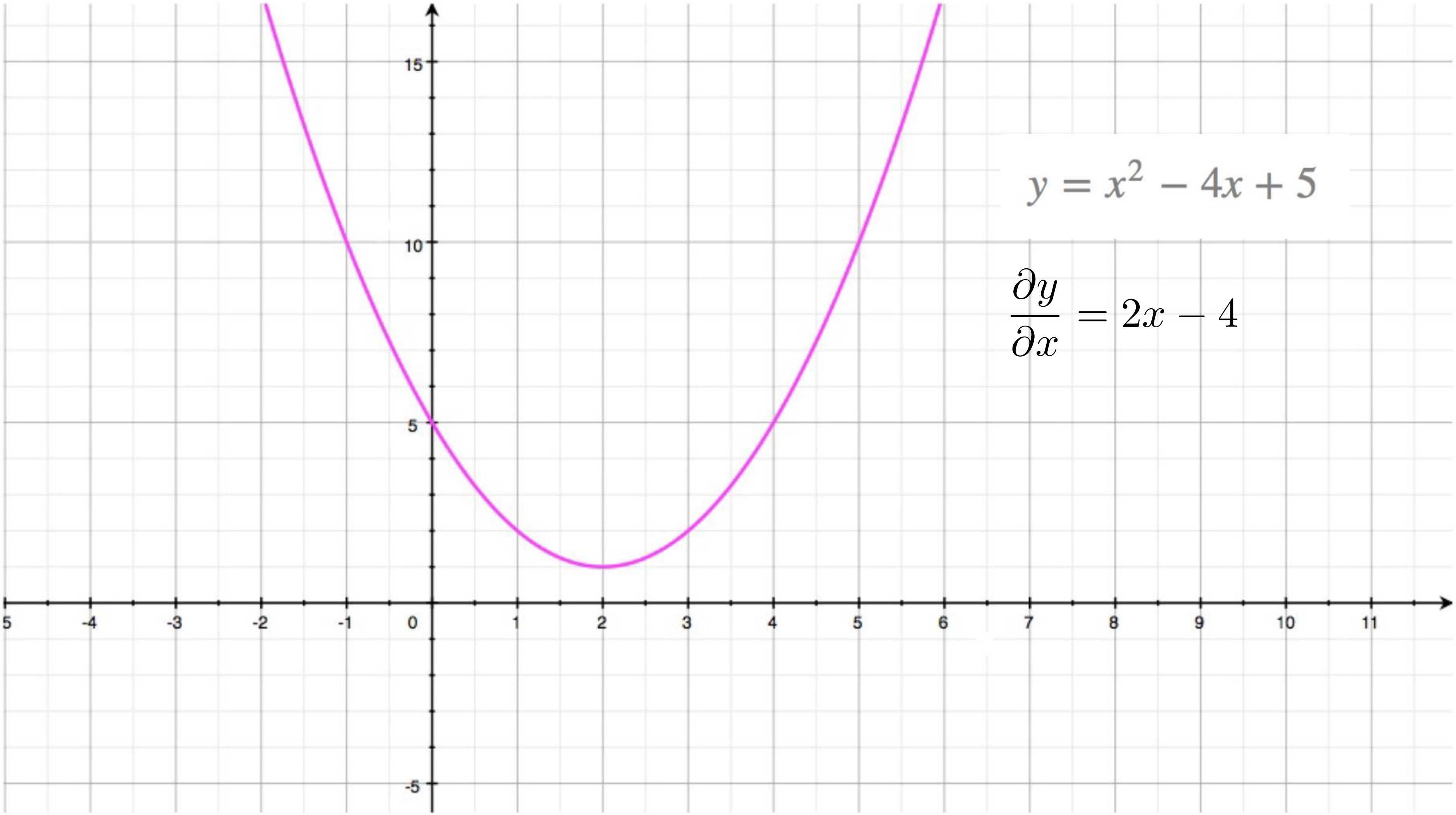
Gradient Descent

knowledgehut upGrad

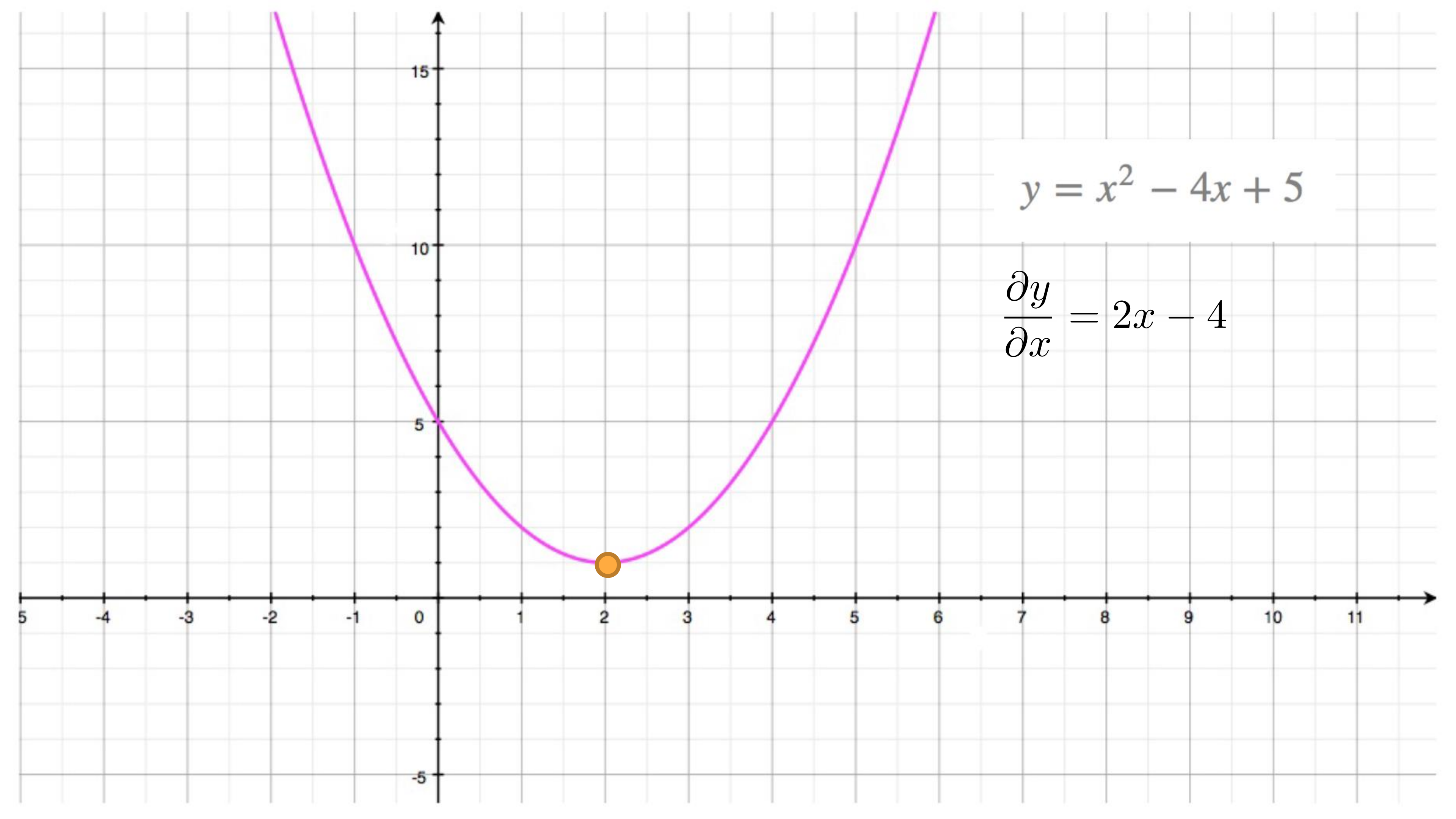
Learning Objectives

- Explain the basics of gradient descent
- Discuss Batch Gradient Descent
- Describe the Stochastic Gradient Descent
- Explain the Mini-batch Gradient Descent

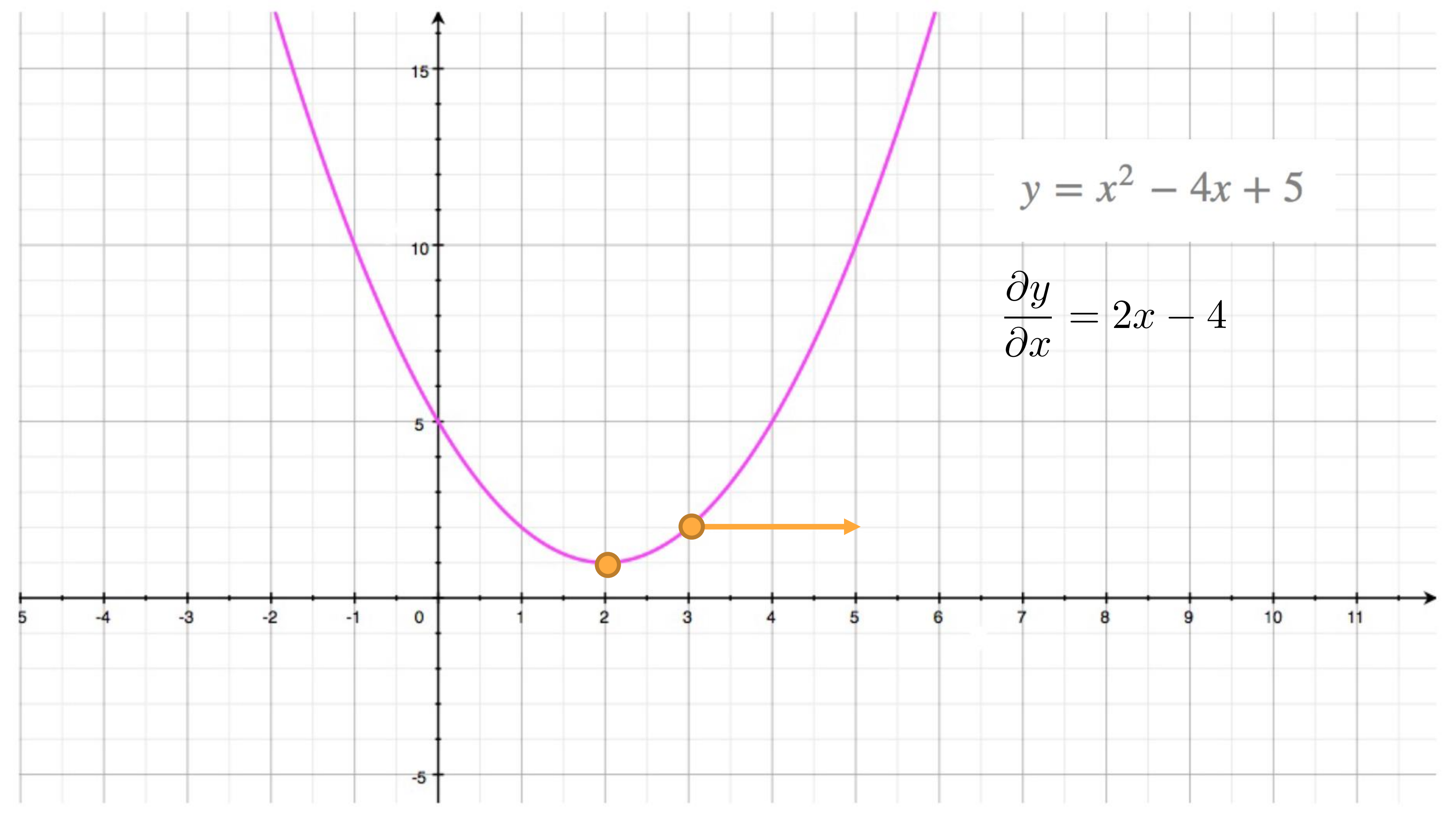
Getting Started

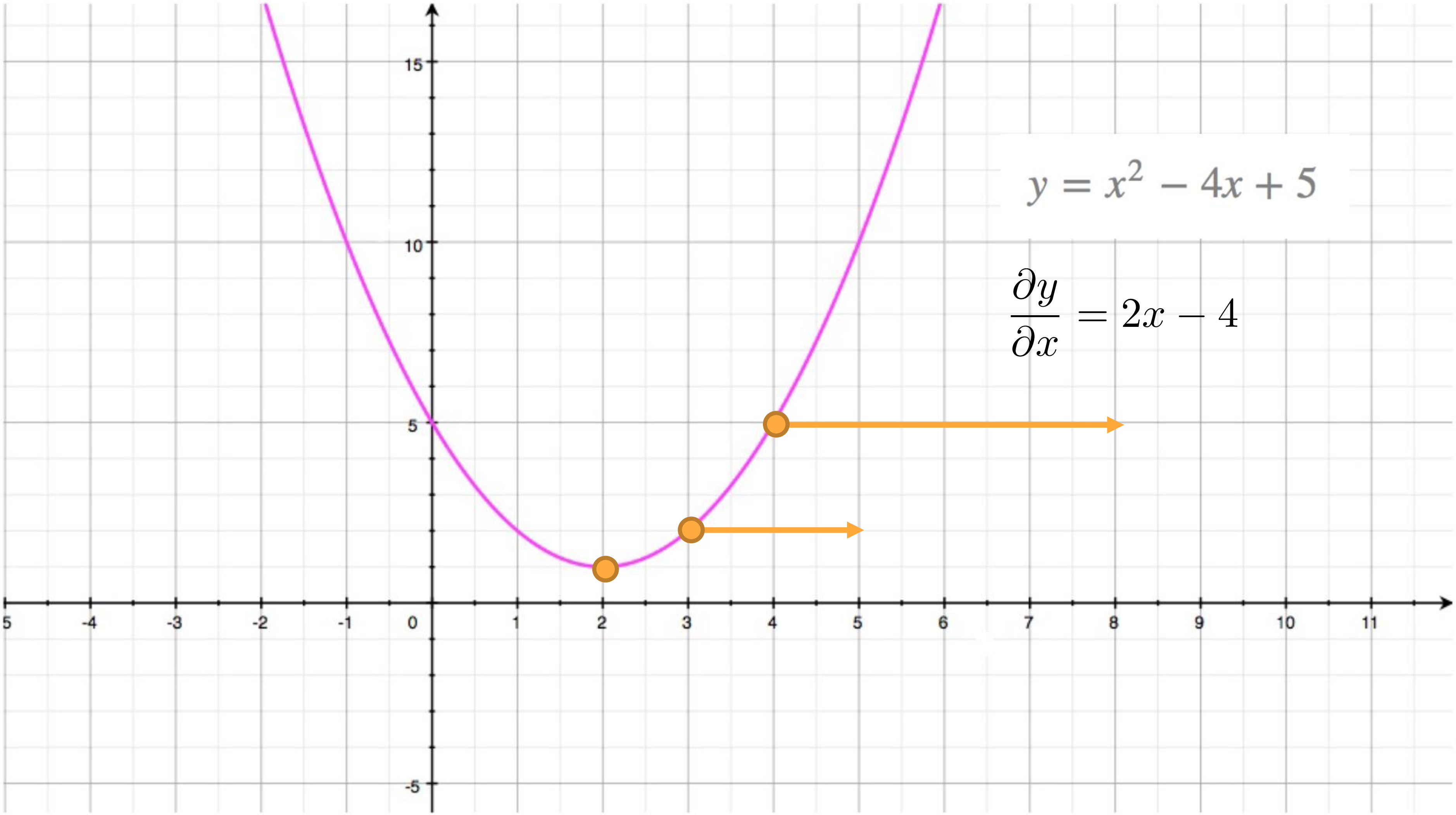


$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
-5
5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11

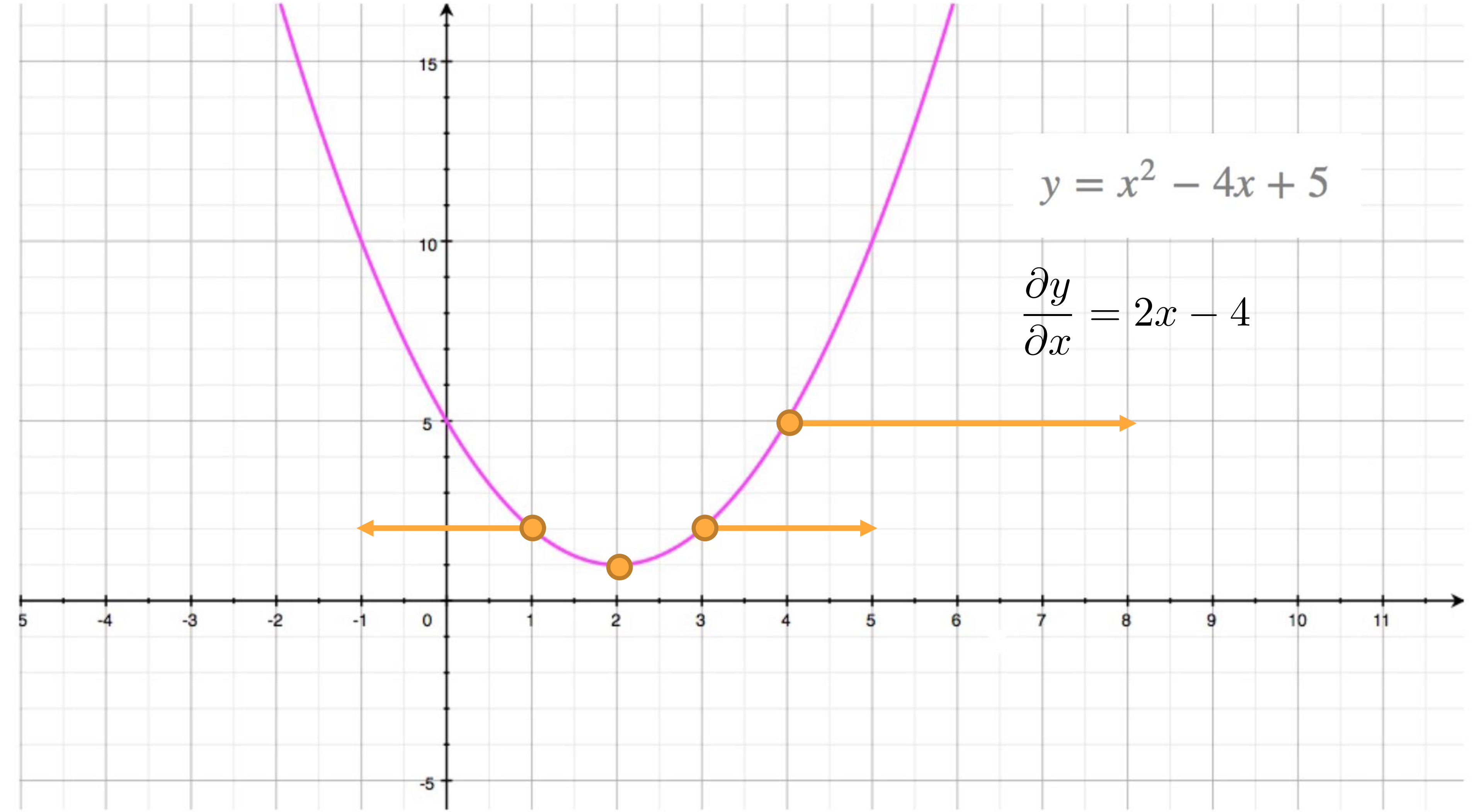


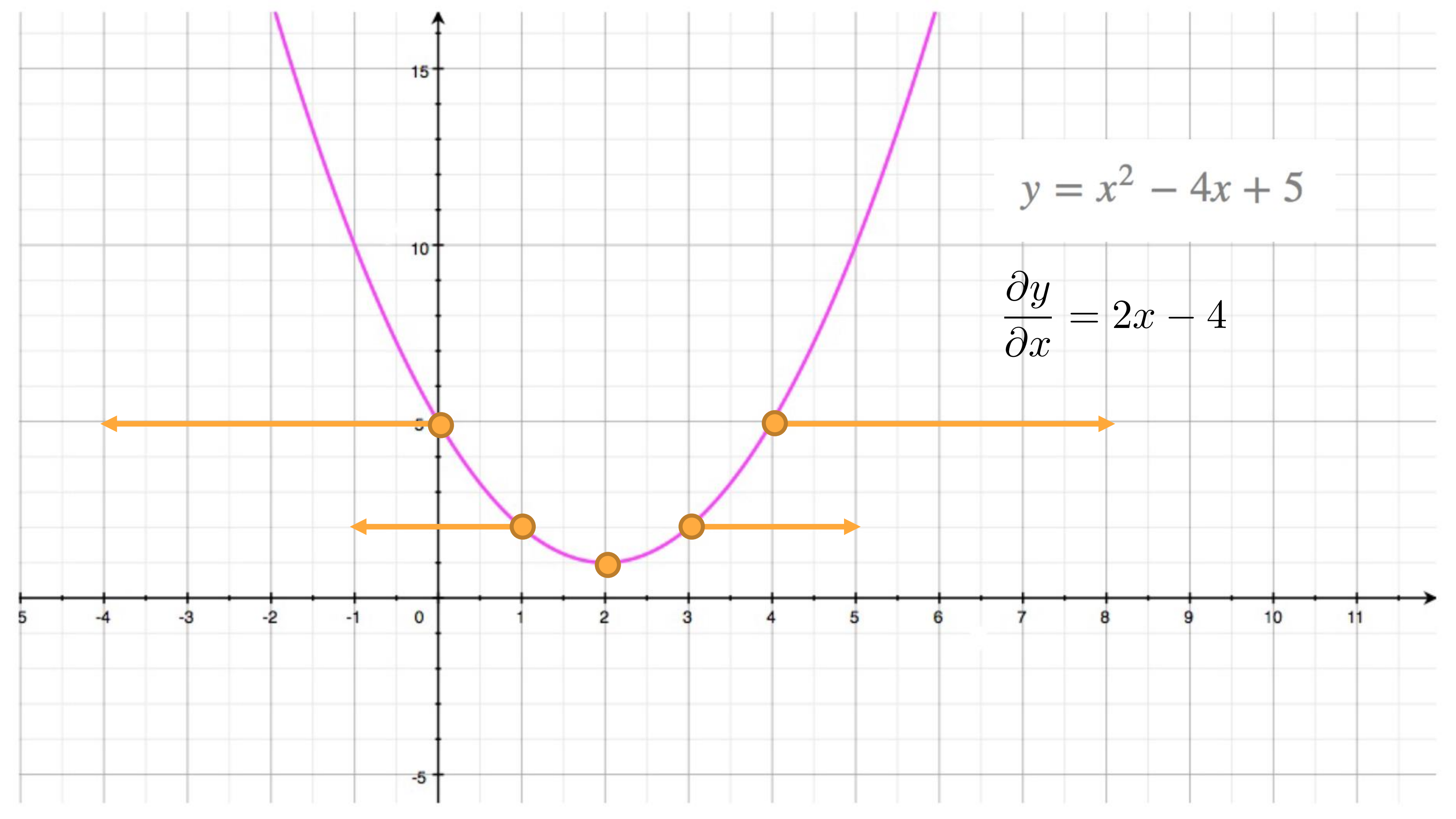
$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
0
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
10
11

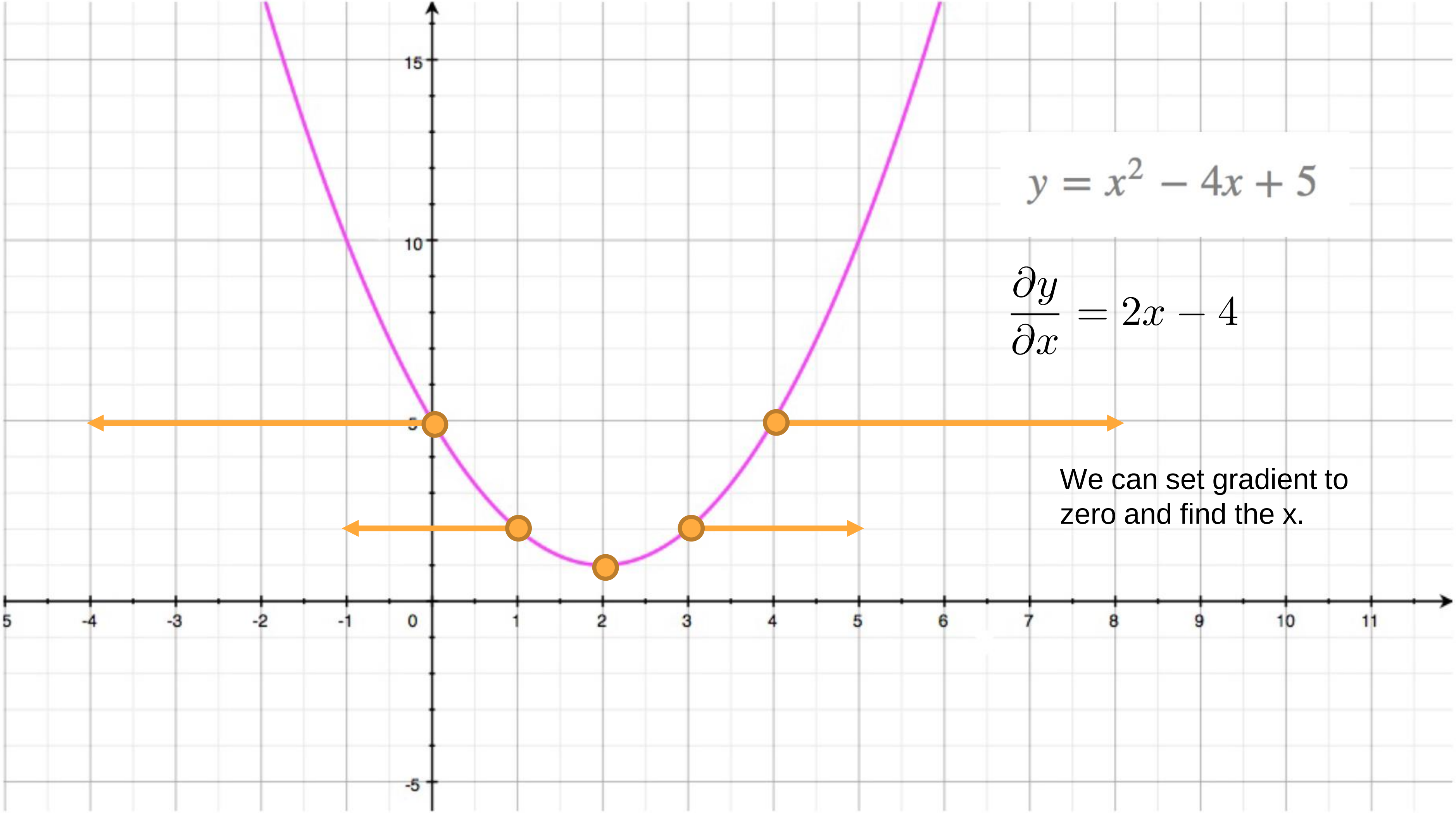

$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
-5
5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11



$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
0
-5
5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11


$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
0
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
10
11

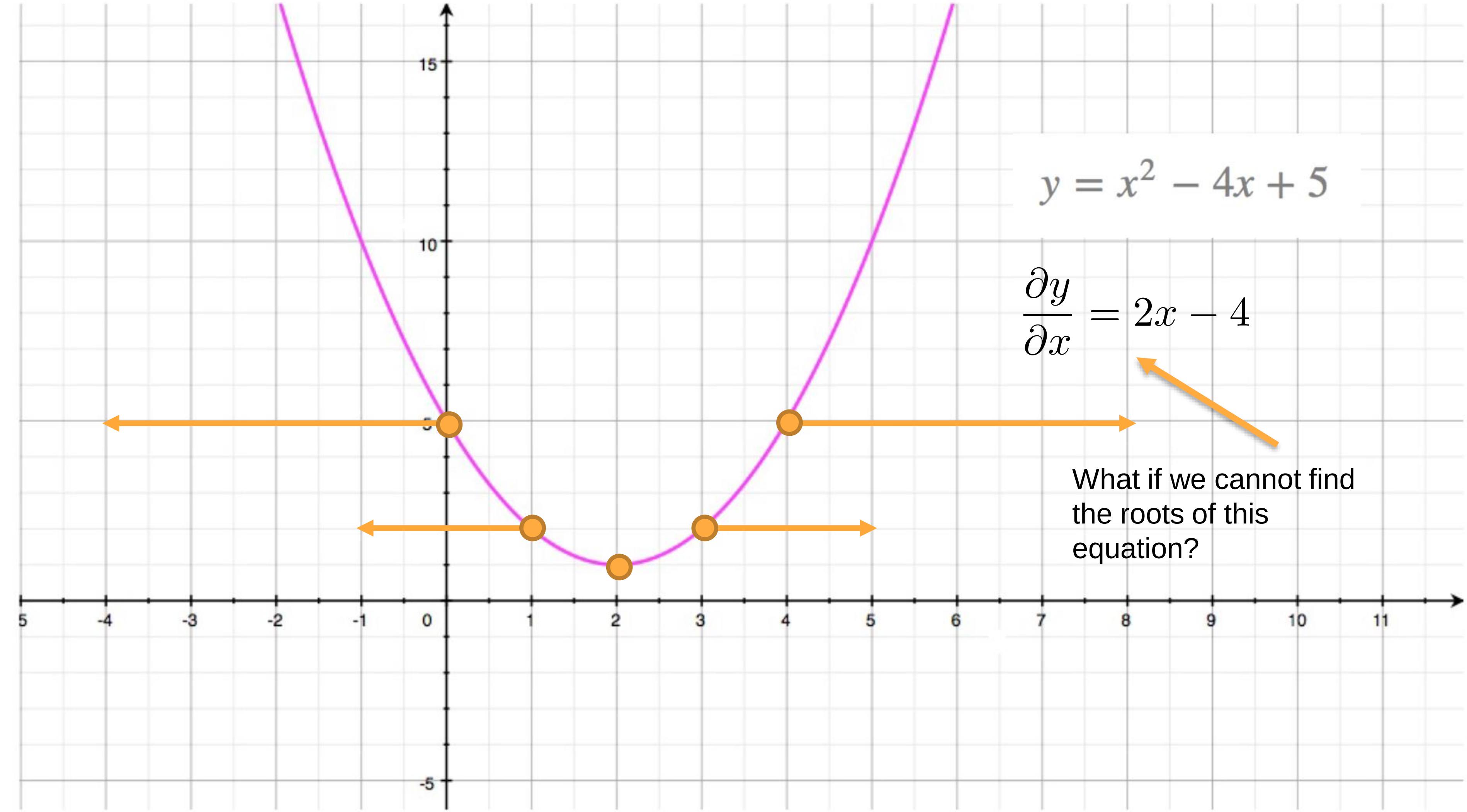

$y = x^2 - 4x + 5$
$\frac{\partial y}{\partial x} = 2x - 4$
15
10
5
-5
5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
10
11



$$y = x^2 - 4x + 5$$

$$\frac{\partial y}{\partial x} = 2x - 4$$

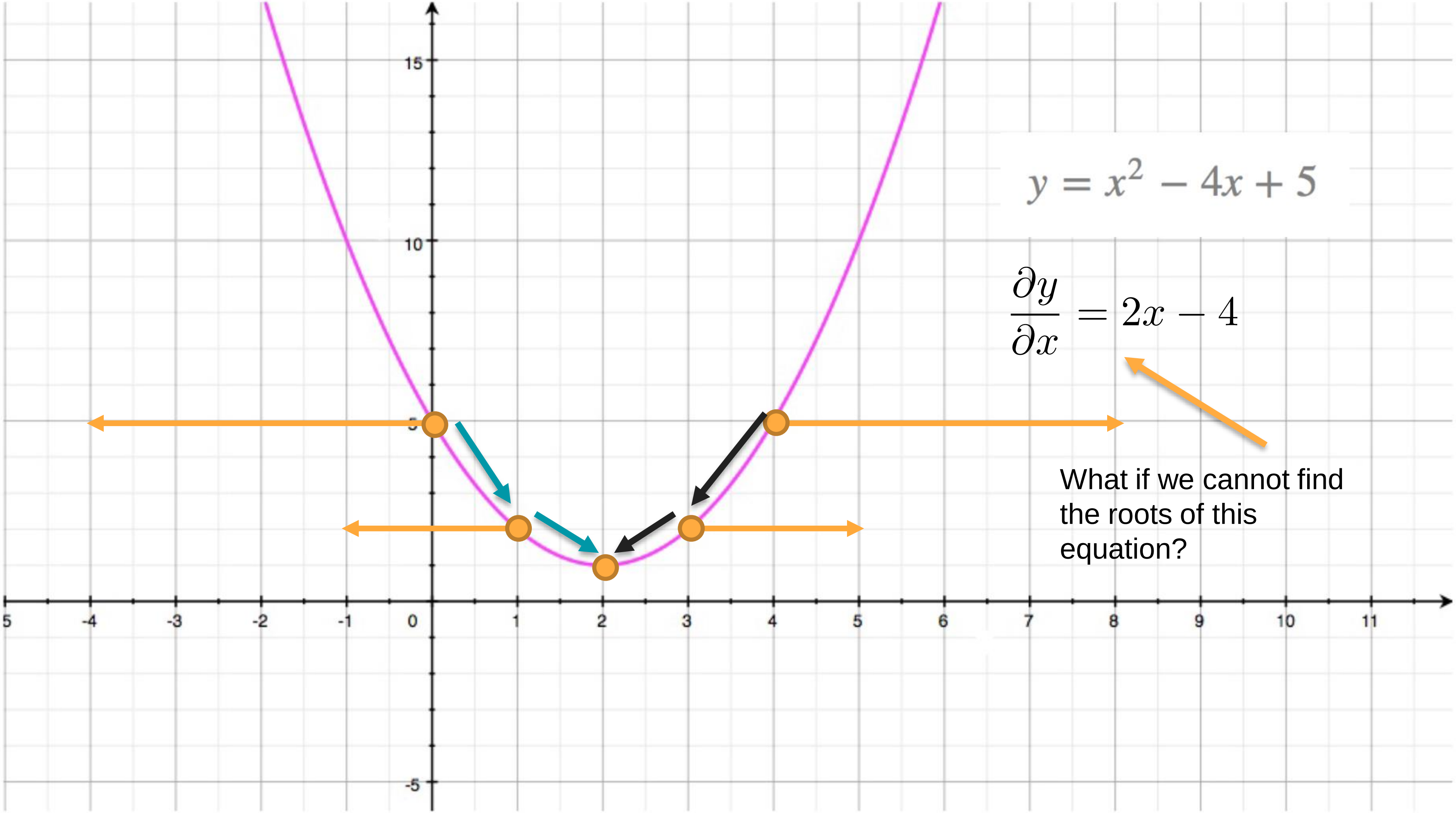
We can set gradient to zero and find the x.



$$y = x^2 - 4x + 5$$

$$\frac{\partial y}{\partial x} = 2x - 4$$

What if we cannot find the roots of this equation?



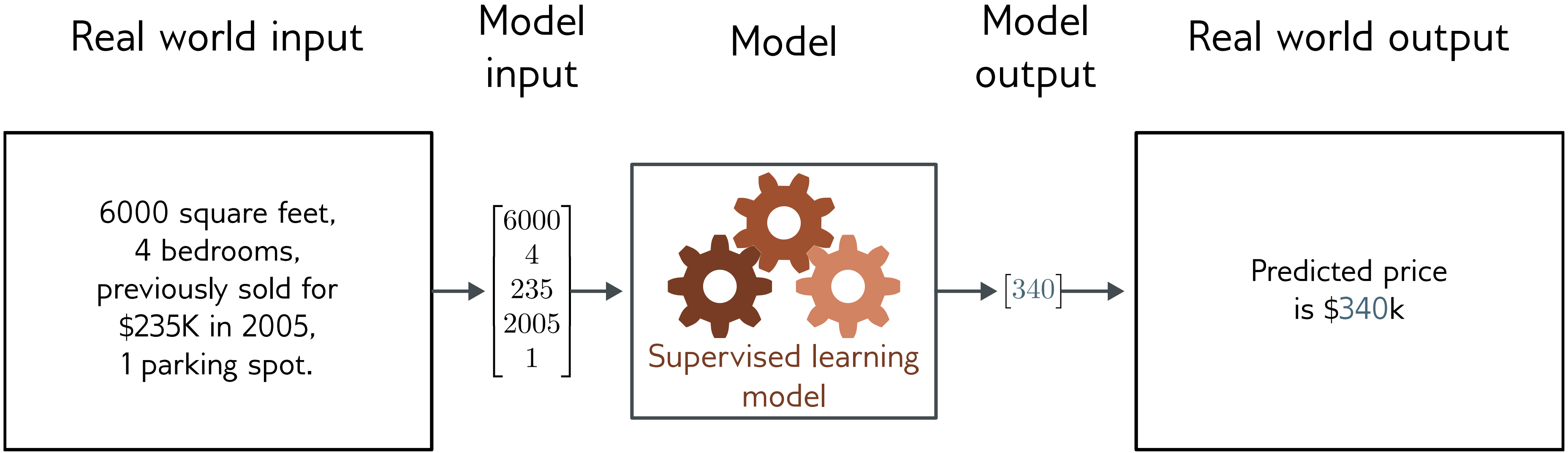
$$y = x^2 - 4x + 5$$

$$\frac{\partial y}{\partial x} = 2x - 4$$

What if we cannot find the roots of this equation?

ML Training vs. Finding the Minima of a Function ?

Regression

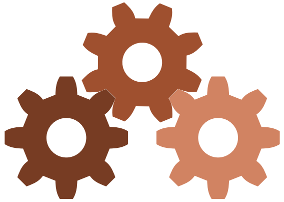


- Univariate regression problem (one output, real value

Text classification

Real world input | Model input | Model | Model output | Real world output

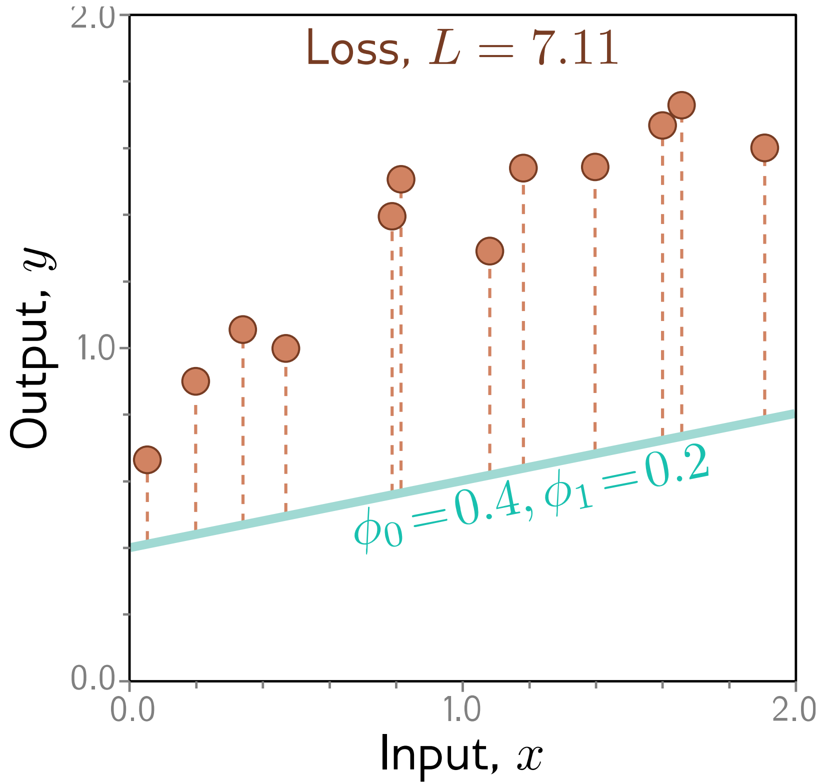
Score → [60] →



Supervised learning model → $\begin{bmatrix} 0.02 \\ 0.98 \end{bmatrix}$ → Failed, Passed

- Binary classification problem (two discrete classes)

1D Linear Regression



Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

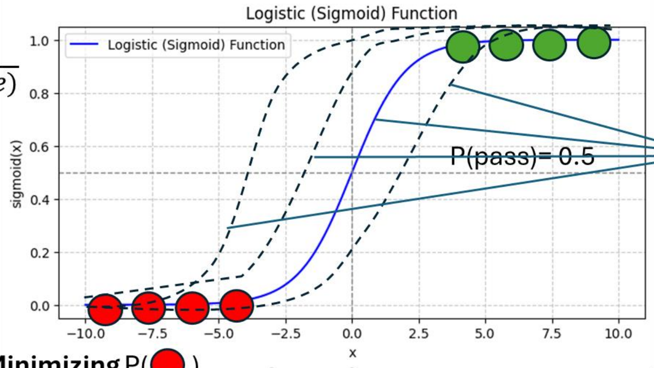
"Least squares loss function"

Logistic Regression

Maximizing P( )

$$\frac{1}{1+e^{-(\beta_0+\beta_1 score)}}$$

P(pass)



Parameterized by

β_0, β_1

Minimizing P( )

$\beta_0 + \beta_1 score$

Or **maximizing** 1 - P( )

From maximization to minimization.

$$L(\beta) = -\prod_{s\ in\ y_i=1} p(x_i) \ * \prod_{s\ in\ y_i=0} (1-p(x_i))$$

Loss function

- Training dataset of I pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

or for short:

$$L[\phi, \mathrm{f}[\mathbf{x}_i, \phi], \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}]$$

Learning parameters, $\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{or}\ \boldsymbol{\varphi}$

Learning model: LR or LG …

$$L[\phi]$$

Returns a scalar that is smaller when model maps inputs to outputs better

Training

- Loss function:

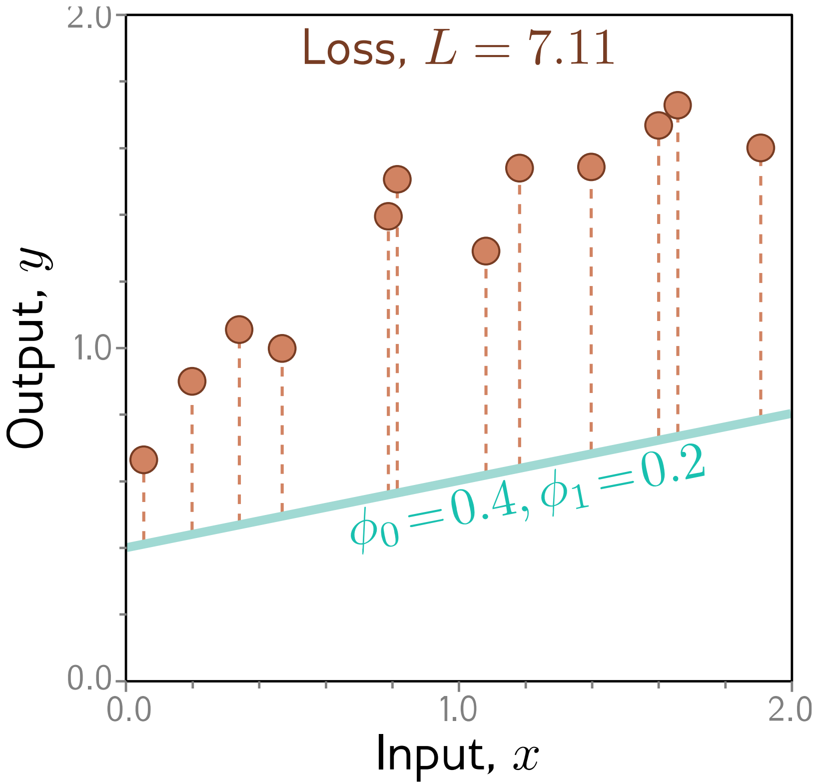
$$L\left[\phi\right]$$

Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\left[L[\phi]\right]$$

Example: 1D Linear regression loss function



Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Example: 1D Linear regression loss function

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[L[\phi] \right]$$

$$\text{minimize } \|y - X\phi\|_2^2$$

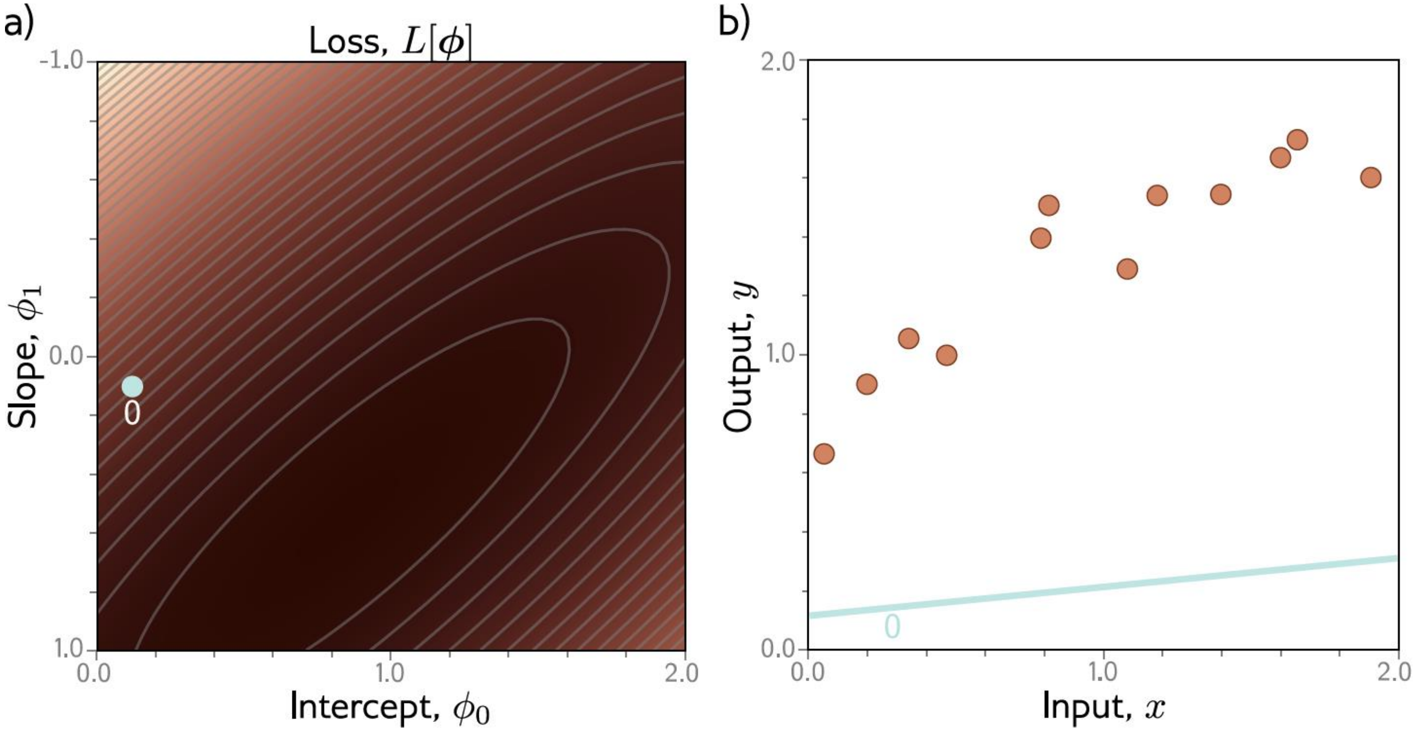
$$\frac{\partial L}{\partial \varphi} = 0 \quad \text{solve for } \hat{\varphi}$$

$$\phi = (X^T X)^{-1} X^T y$$ → Closed-form solution

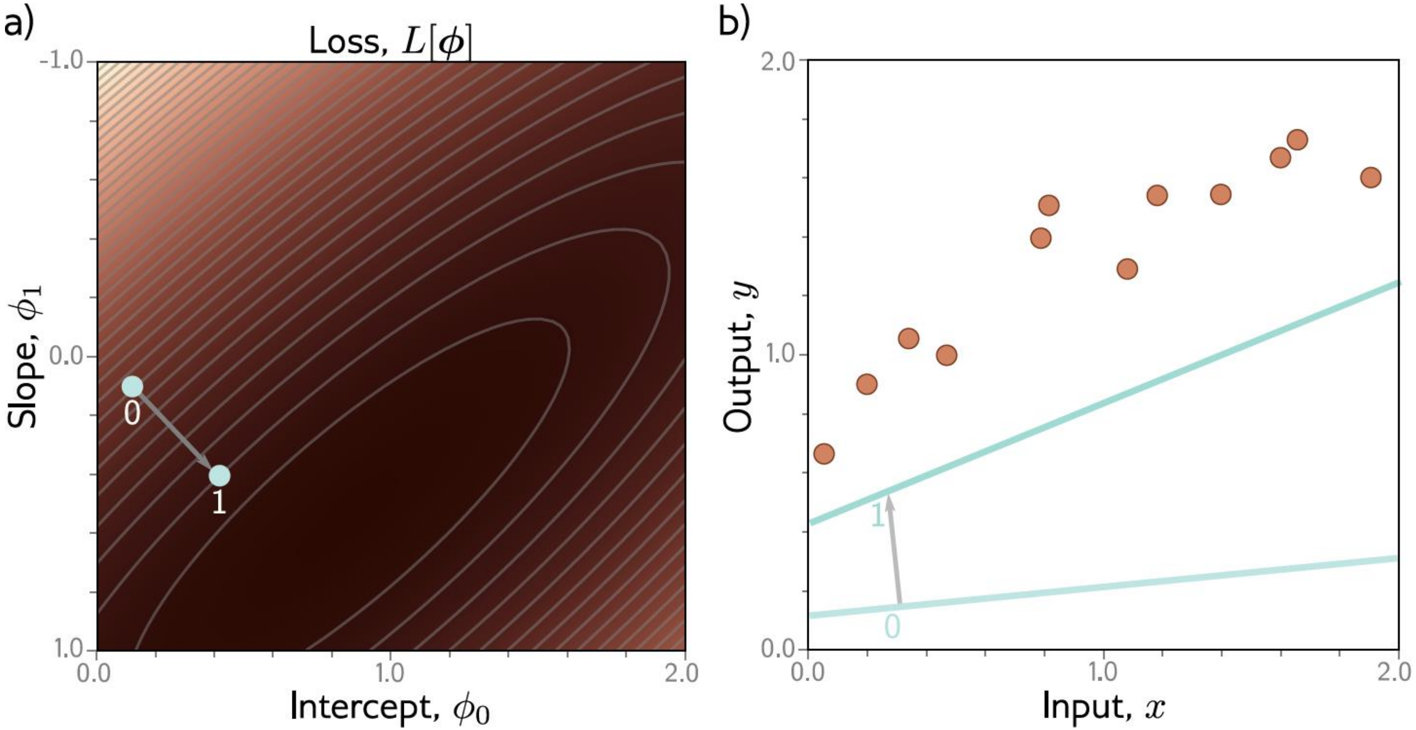
Where:

- $(X^T X)^{-1}$ is the inverse of the matrix product of the transpose of the design matrix X^T and X.
- X^T is the transpose of the design matrix.
- y is the vector of observed target values.

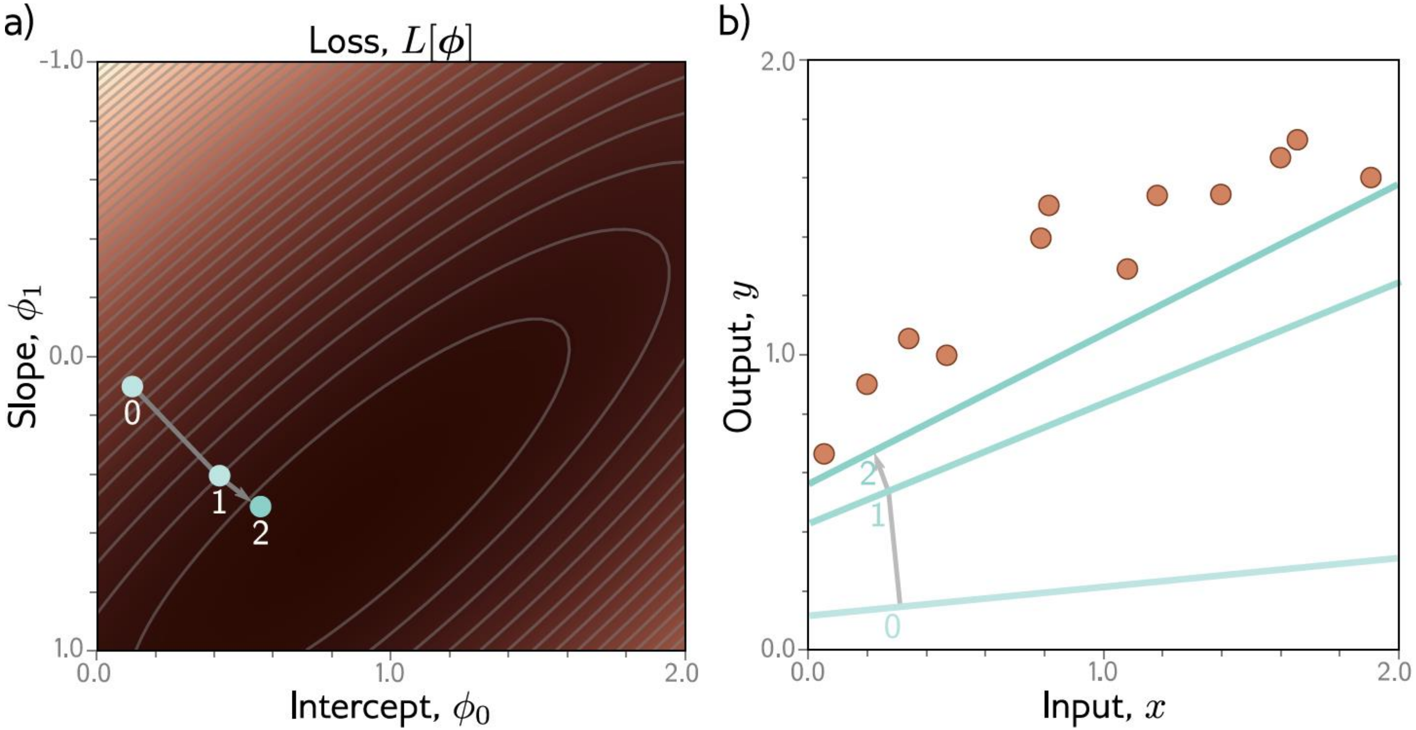
What if there is no closed-form solution ?



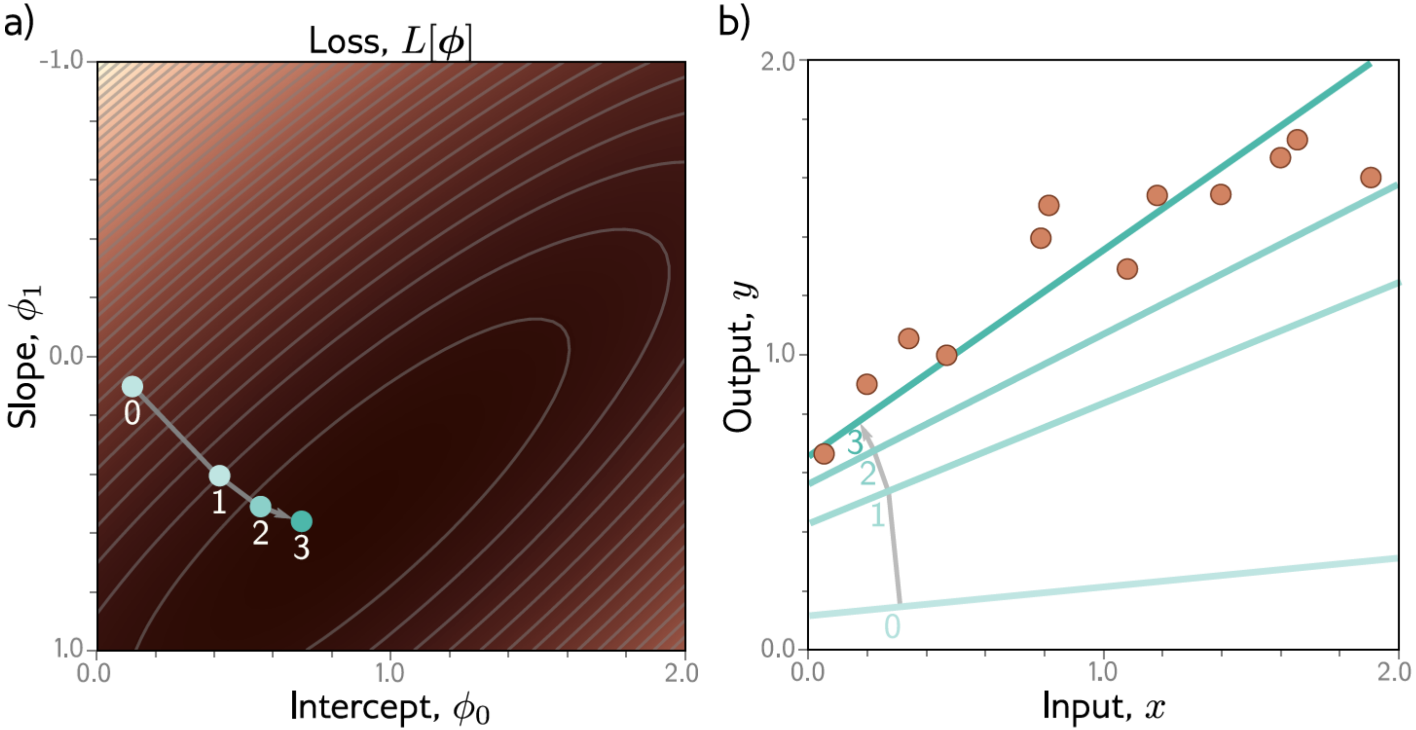
Example: 1D Linear regression training



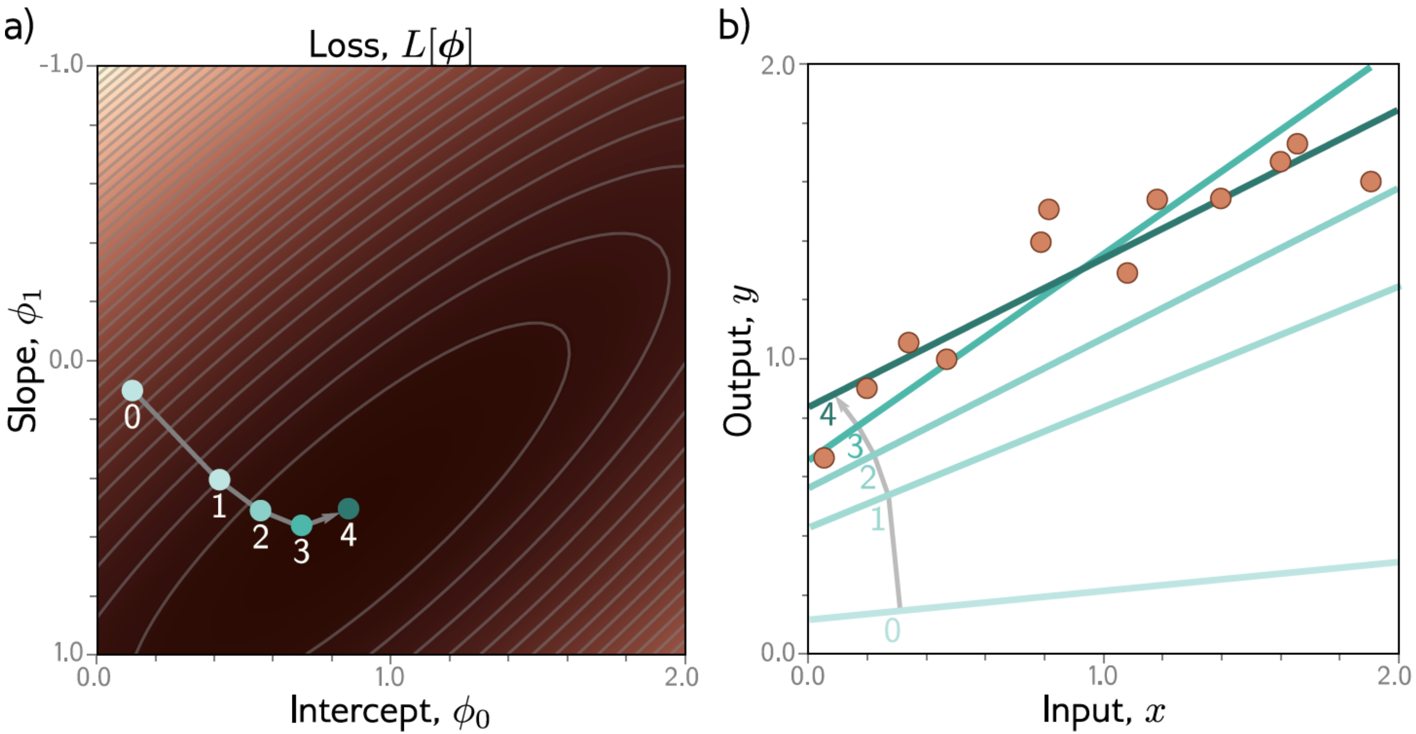
Example: 1D Linear regression training



Example: 1D Linear regression training



Example: 1D Linear regression training



This technique is known as gradient descent

Introduction to Gradient Descent

Gradient descent algorithm

Step 1. Compute the derivatives of the loss with respect to the parameters:

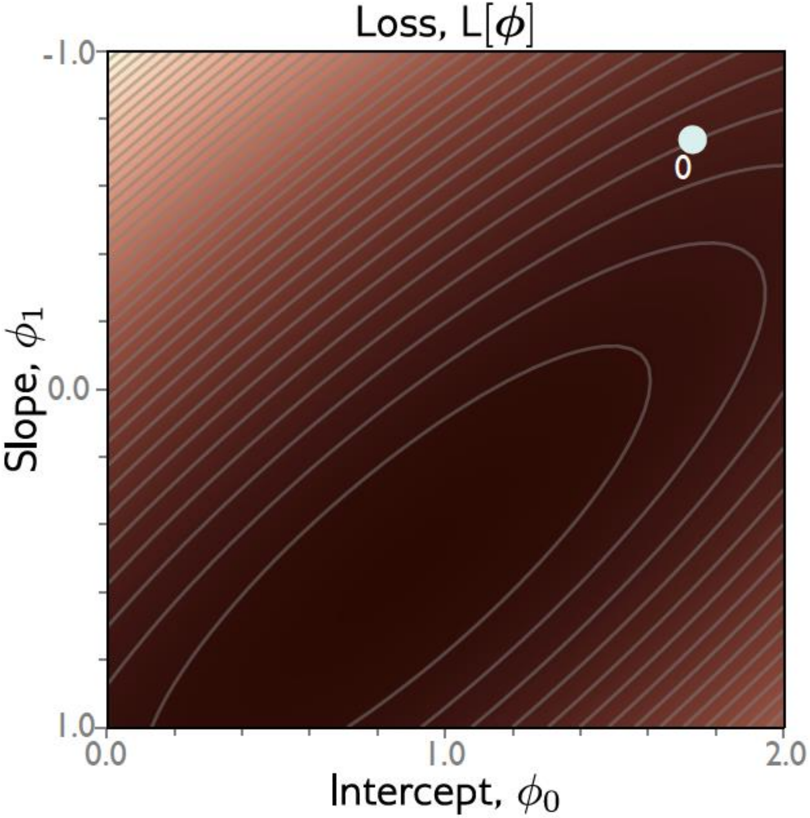
$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial L}{\partial \phi_0} \\ \frac{\partial L}{\partial \phi_1} \\ \vdots \\ \frac{\partial L}{\partial \phi_N} \end{bmatrix}.$$

Step 2. Update the parameters according to the rule:

$$\boldsymbol{\phi} \longleftarrow \boldsymbol{\phi} - \alpha \frac{\partial L}{\partial \boldsymbol{\phi}},$$

where the positive scalar α determines the magnitude of the change.

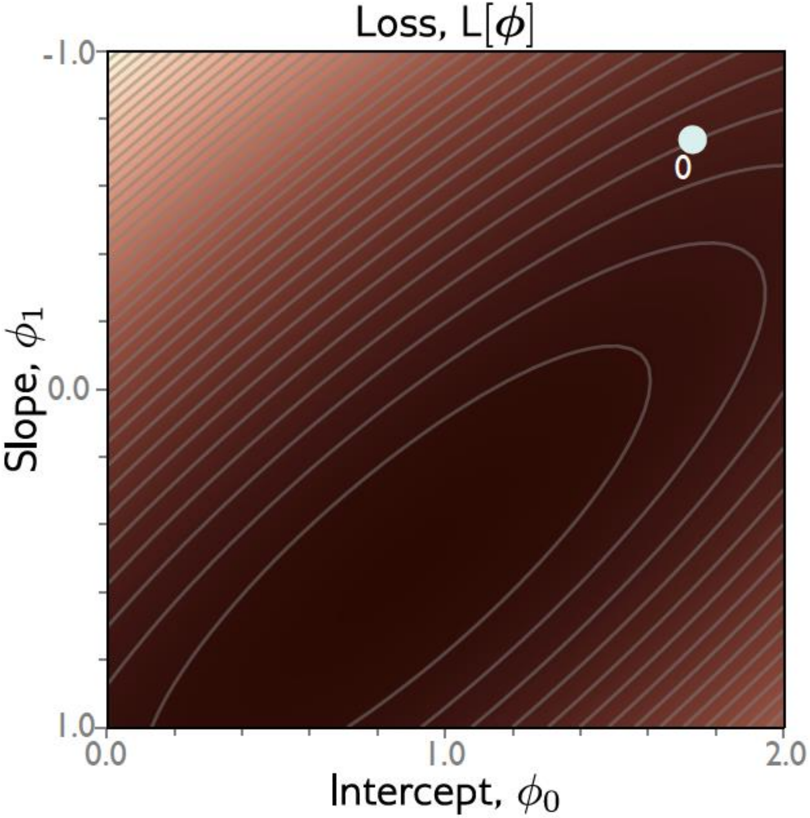
Gradient descent



Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

Gradient descent

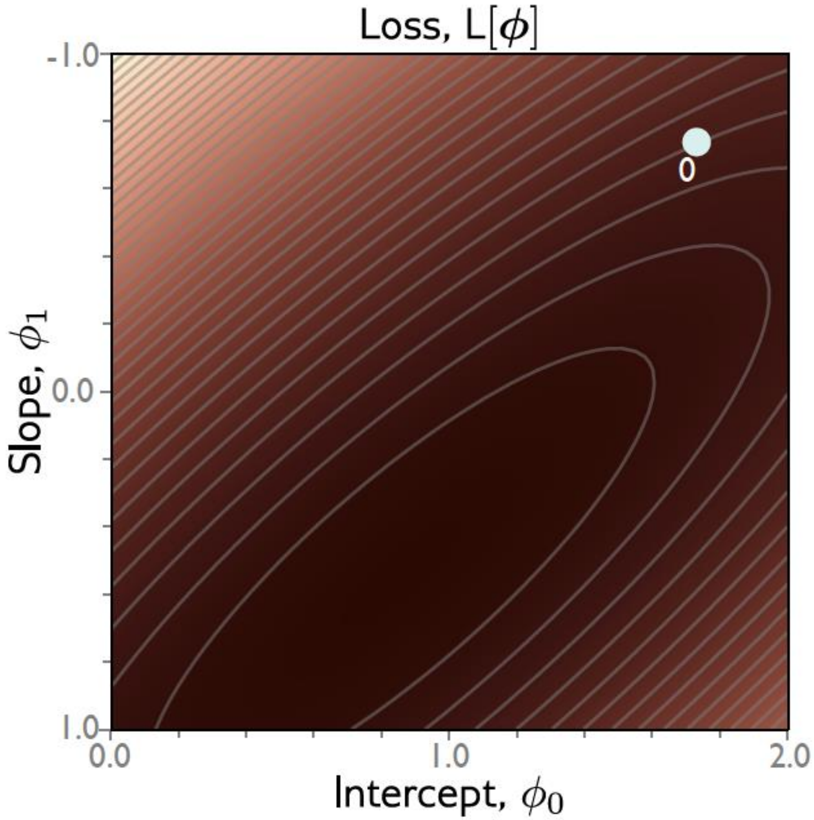


Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

$$
\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}
$$

Gradient descent



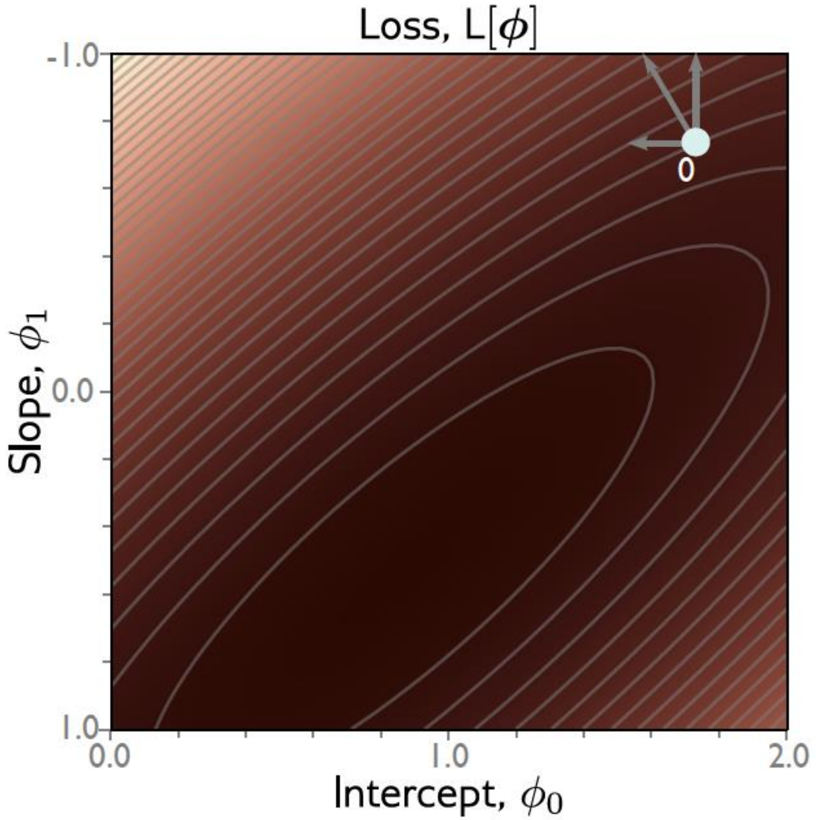
Step 1: Compute derivatives (slopes of function) with
Respect to the parameters

$$
\begin{aligned}
L[\boldsymbol{\phi}] &= \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \left(\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i\right)^2 \\
&= \sum_{i=1}^{I} \left(\phi_0 + \phi_1 x_i - y_i\right)^2
\end{aligned}
$$

$$
\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}
$$

$$
\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}
$$

Gradient descent

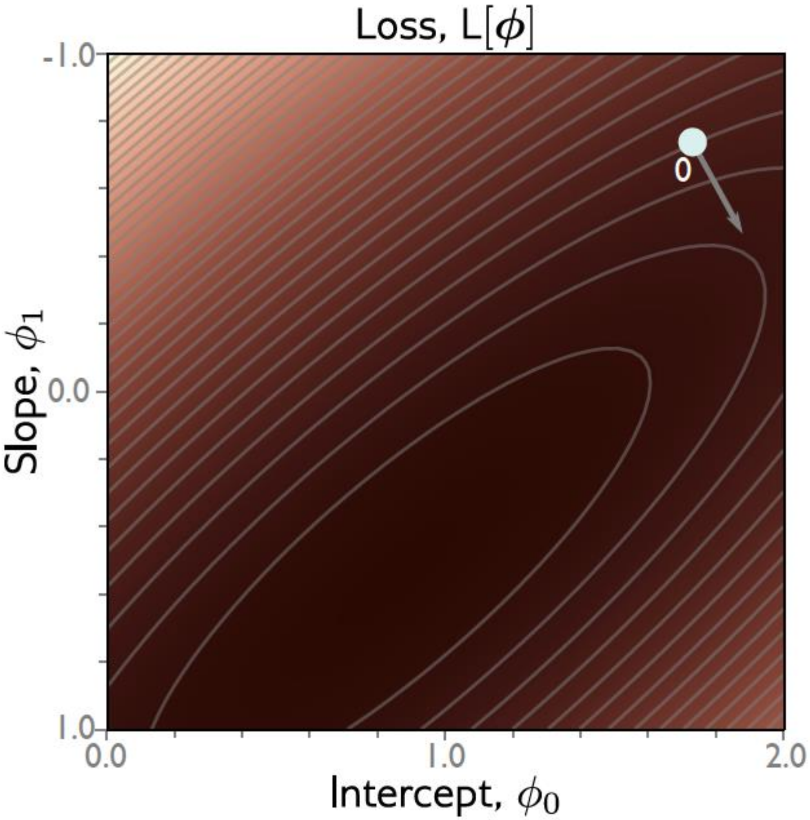


Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

$$\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Gradient descent



Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi}\sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \phi}$$

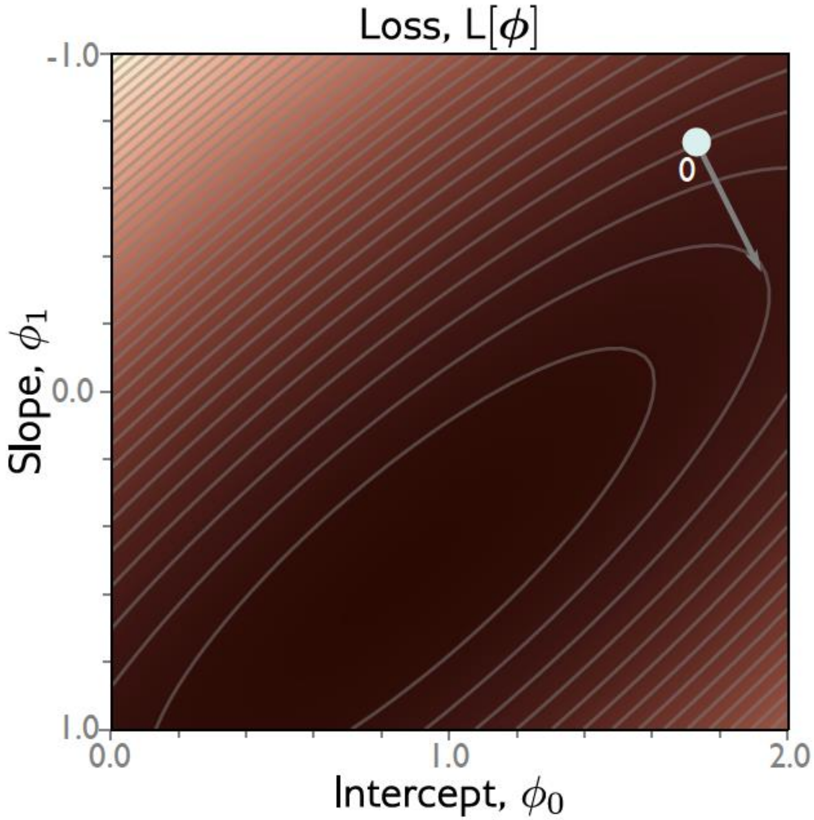
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

α = step size or learning rate if fixed

Gradient descent



Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \phi}$$

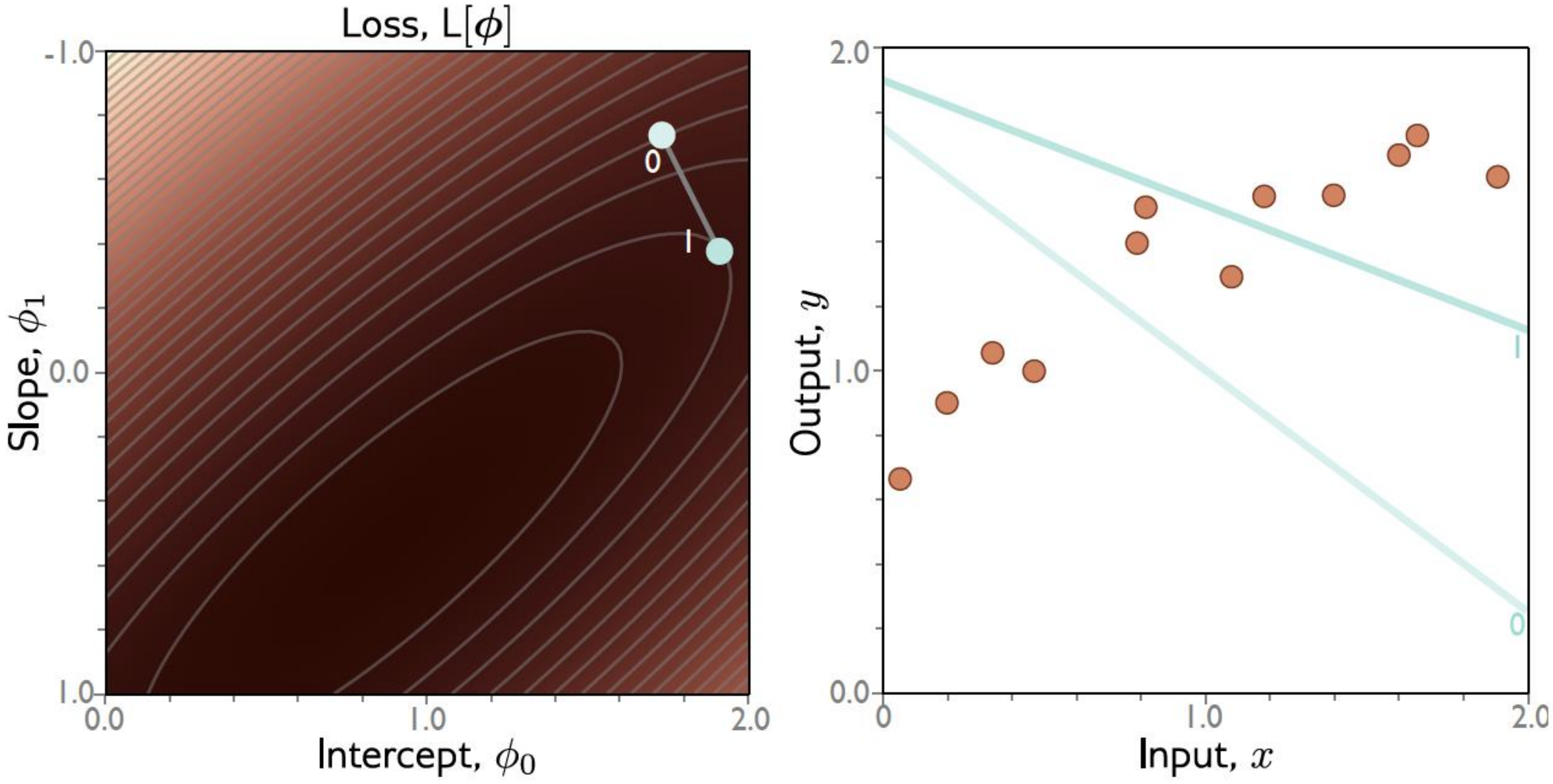
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

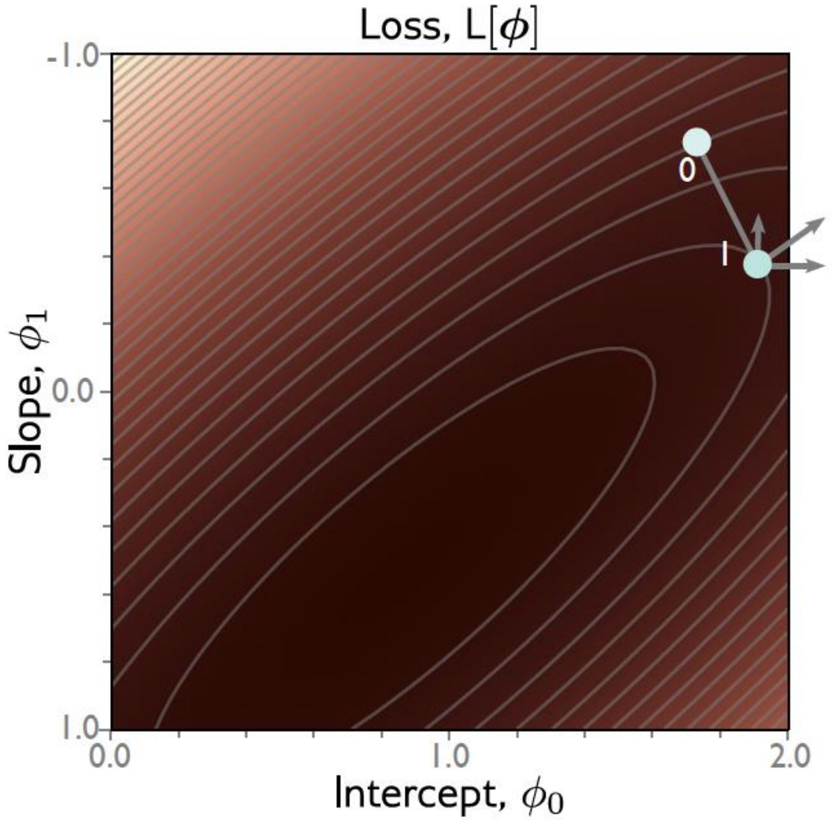
$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

α = step size

Gradient descent



Gradient descent



Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi}\sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \phi}$$

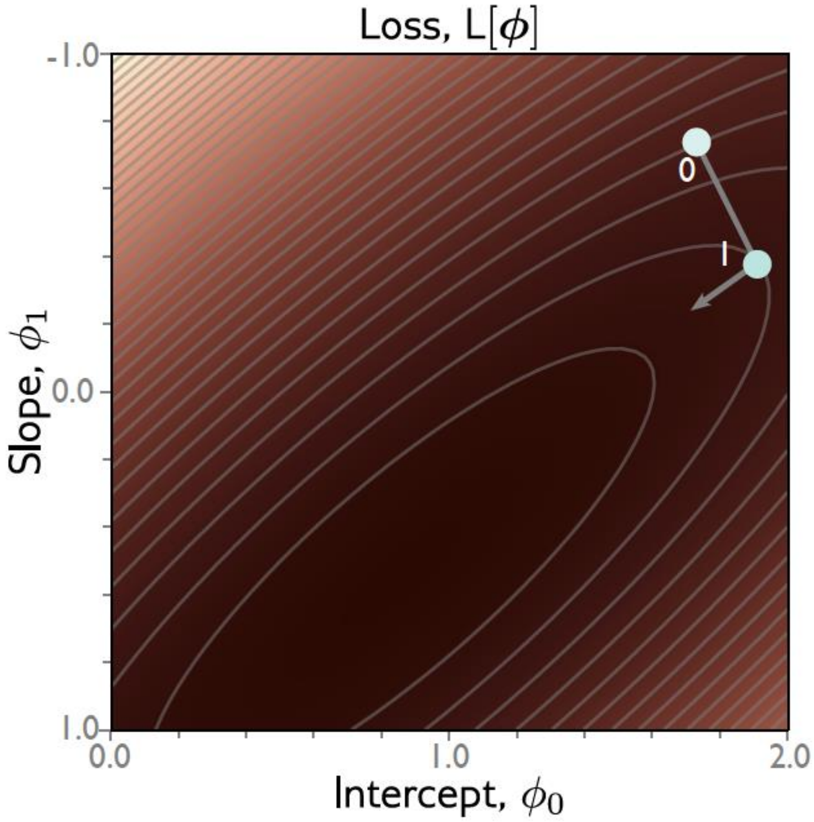
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

α = step size

Gradient descent



Step 1: Compute derivatives (slopes of function) with Respect to the parameters

$$\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \phi}$$

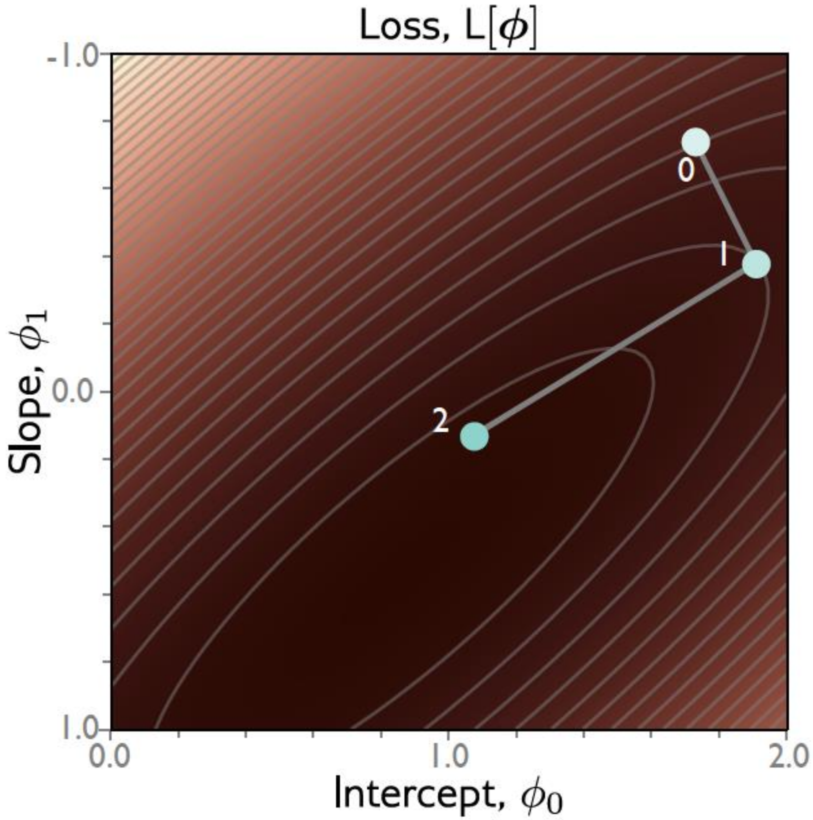
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

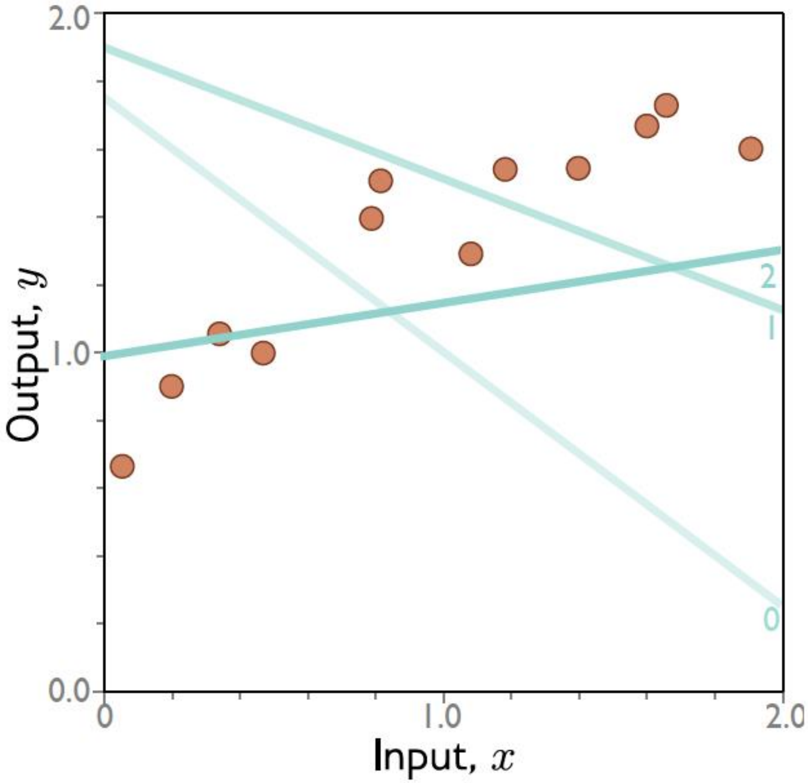
Step 2: Update parameters according to rule

$$\phi \longleftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

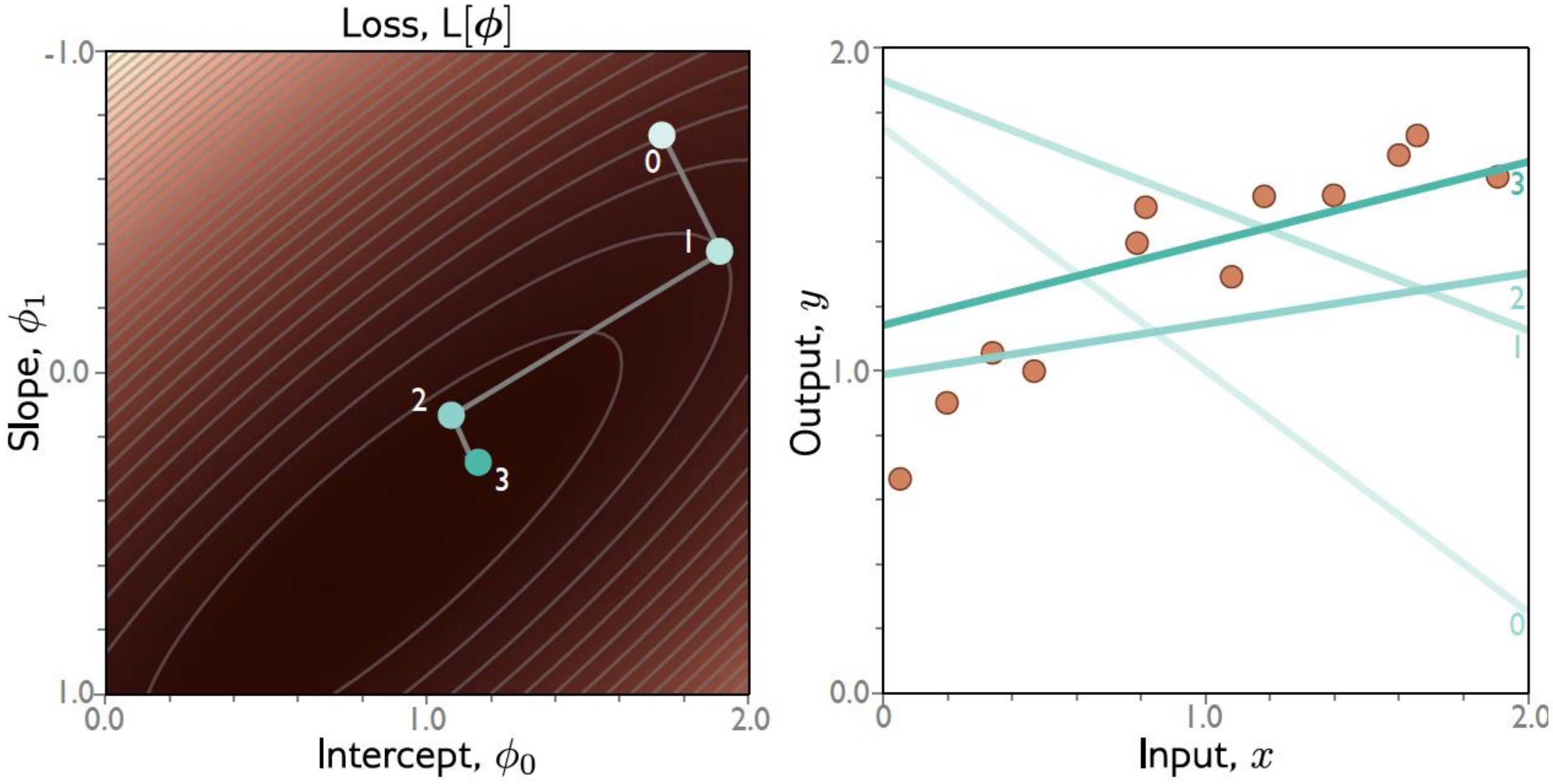
α = step size

Gradient descent

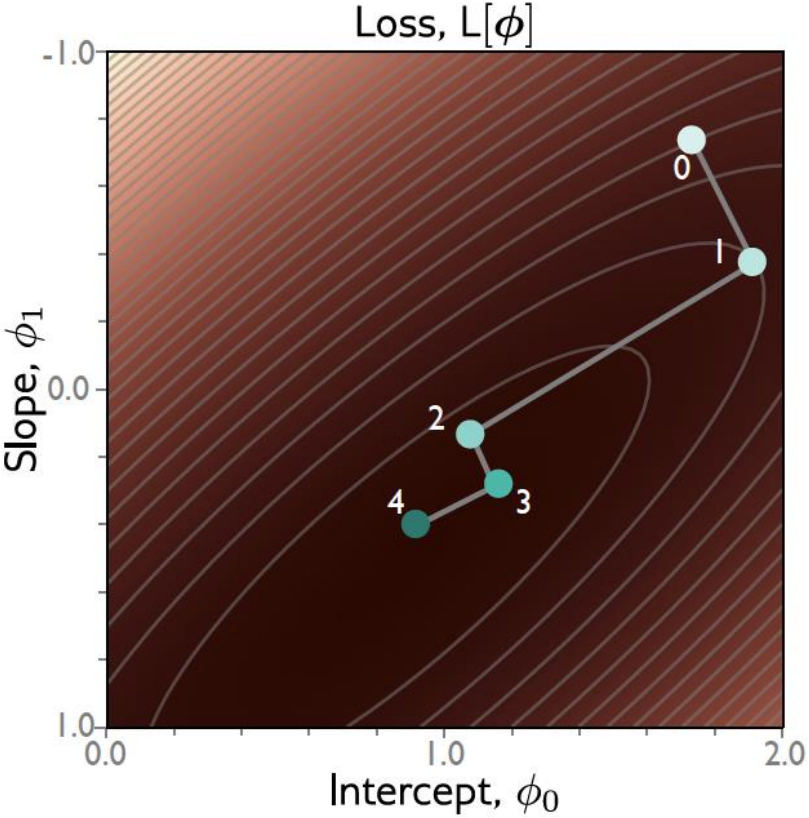


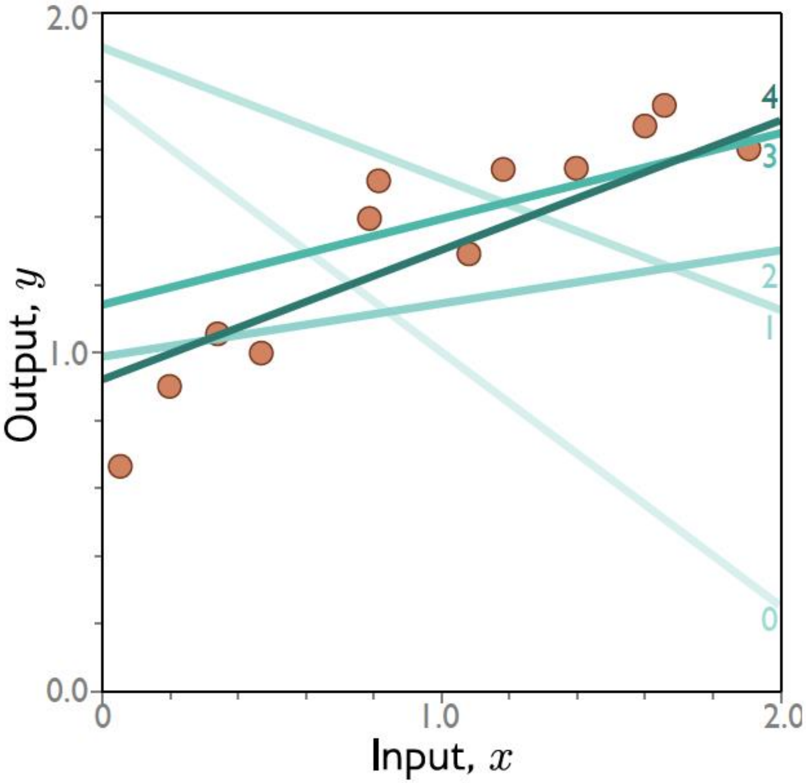


Gradient descent

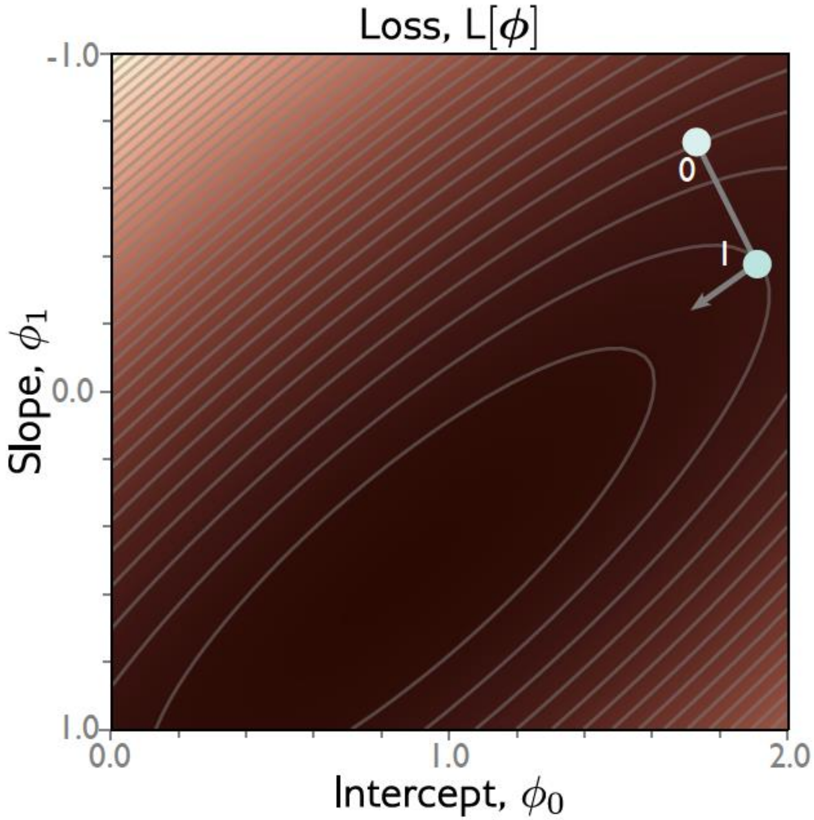


Gradient descent





Line Search



Step 1: Compute derivatives (slopes of function) with
Respect to the parameters

$$\frac{\partial L}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} \sum_{i=1}^{I} \ell_i = \sum_{i=1}^{I} \frac{\partial \ell_i}{\partial \boldsymbol{\phi}}$$

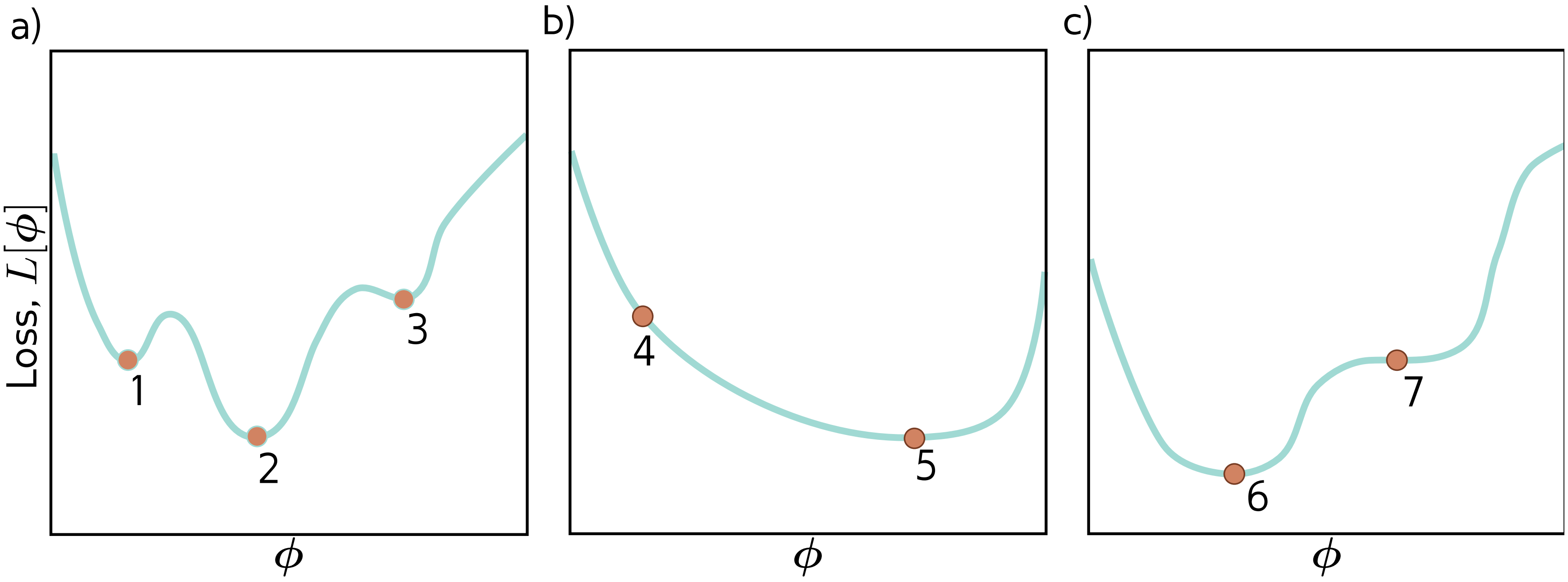
$$\frac{\partial \ell_i}{\partial \boldsymbol{\phi}} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

$$\boldsymbol{\phi} \longleftarrow \boldsymbol{\phi} - \alpha \frac{\partial L}{\partial \boldsymbol{\phi}}$$

α = step size

Convex problems

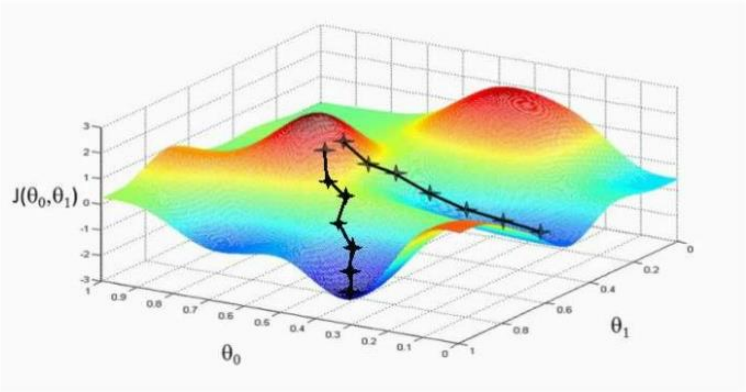


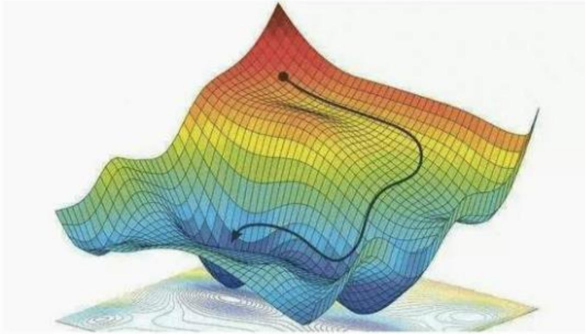
Non convex

Convex

Non-Convex

How Does Gradient Descent Work?

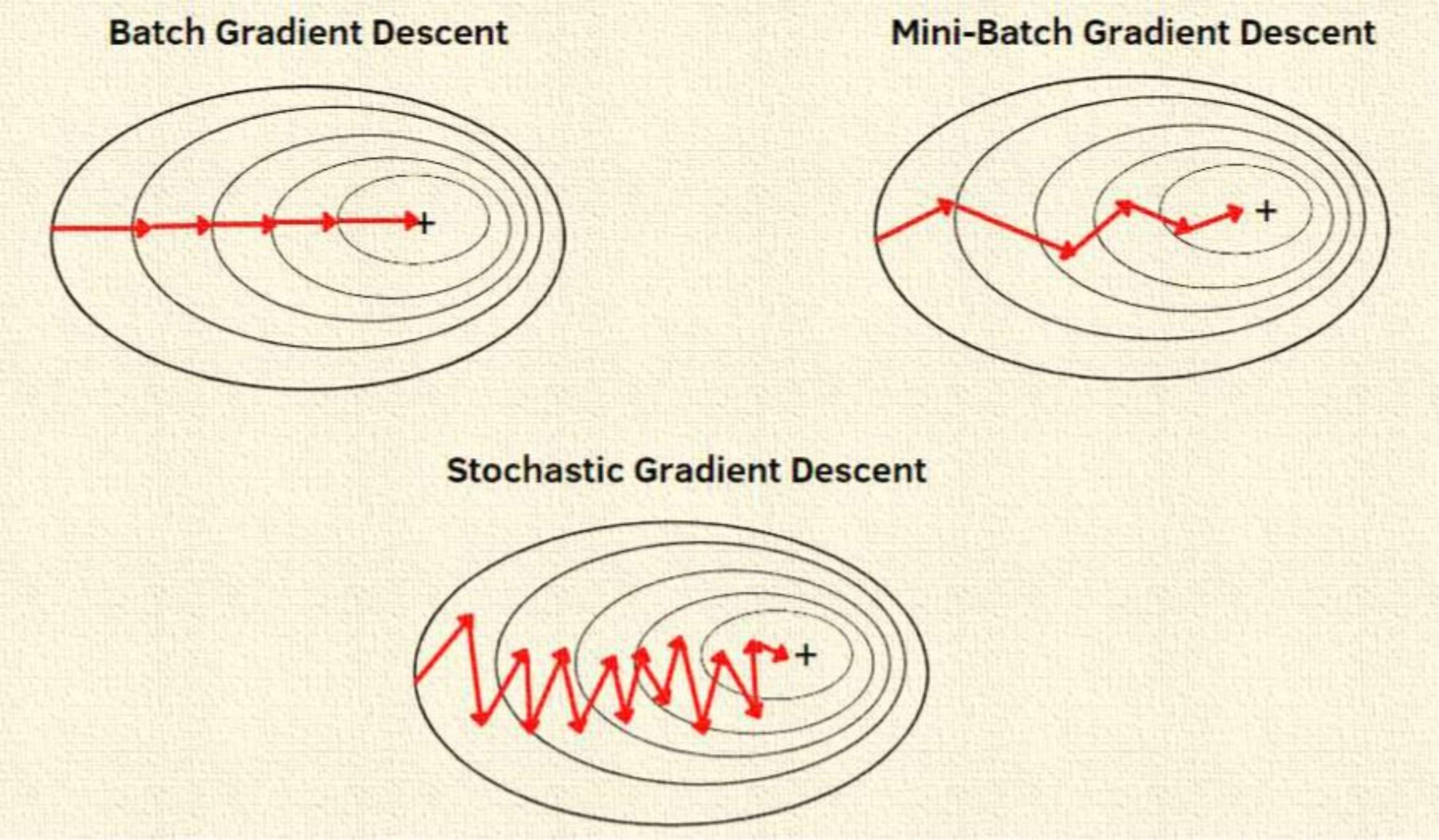


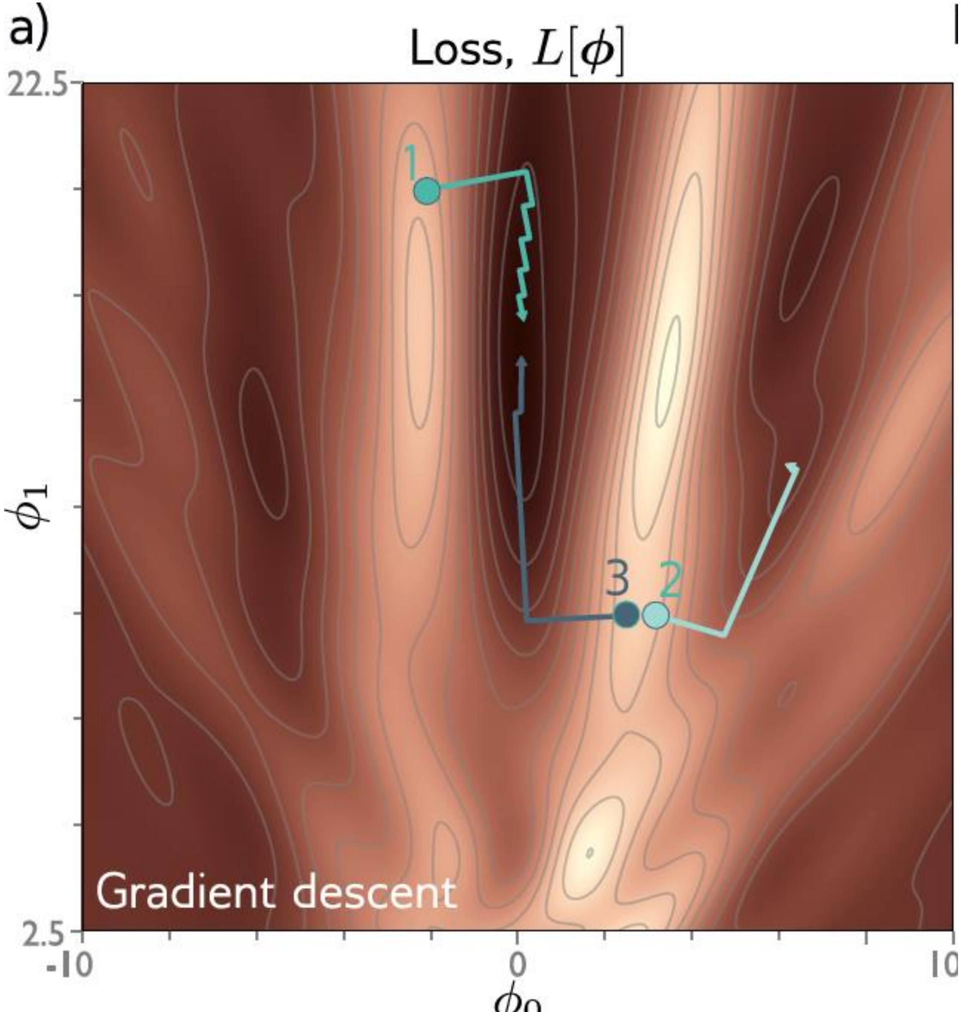


https://en.wikipedia.org/wiki/Gradient_descent

- Having more than one input variable or a more complex function than the Squared Loss, one might get Loss Functions that look like landscapes.
- Gradient Descent will help find the minimum of these functions by cleverly combining different combinations of parameters.

Gradient Descent Types





Batch Gradient Descent

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i=1}^{I} \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

Mini-Batch Gradient Descent

$$\phi_{t+1} \longleftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

Stochastic Gradient Descent

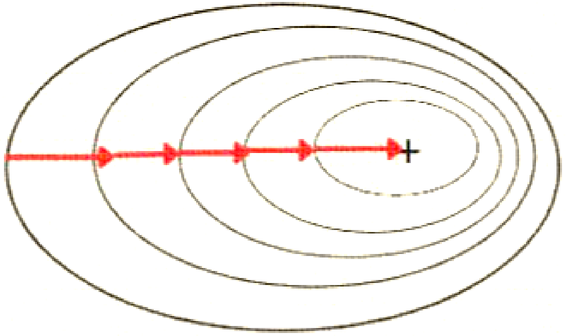
Batch size is 1.

Fixed learning rate α

Batch Gradient Descent

Batch Gradient Descent

Batch Gradient Descent is also called Vanilla Gradient Descent because of its pure, unmodified version of a vanilla version.



Pros

- Produces a stable error gradient and a stable convergence
- Deterministic (given the same data)

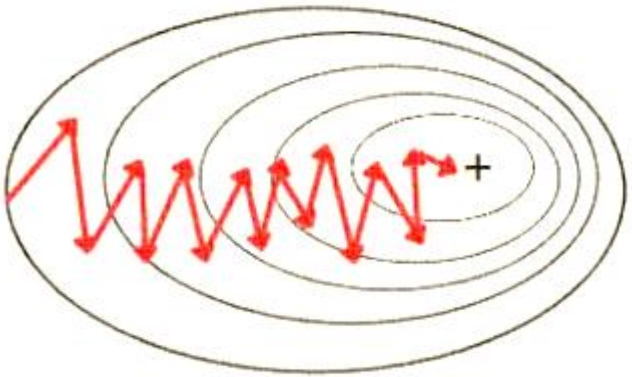
Cons

- This can be problematic for very large datasets
- Can converge in a way that is not optimal (get stuck in local minimum)

Stochastic Gradient Descent

Stochastic Gradient Descent

The algorithm will update the weights after each training example. To make this method work, the dataset needs to be shuffled, and the algorithm needs to take random training samples.



Pros

- Works for very large datasets
- Can potentially converge faster than Batch Gradient Descent
- Chances of being trapped in local minimum are lower

Cons

- Path to a minimum can be noisy
- Non-deterministic

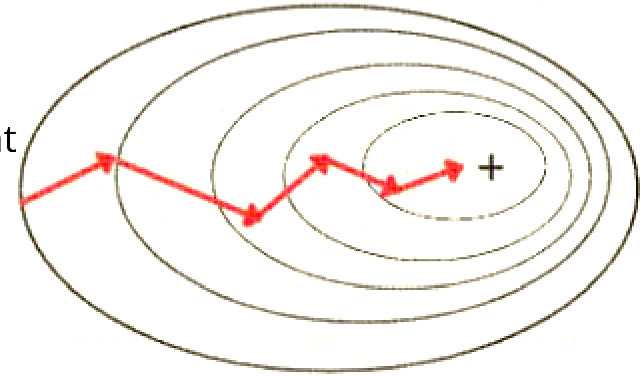
Mini-batch Gradient Descent

Mini-batch Gradient Descent (Standard Practice)

The algorithm will update the weights after the n training example. This dataset is usually shuffled as well as in Stochastic Gradient Descent.

Pros

- Compromise between Batch and Stochastic Gradient descent
- Batch size can be adapted depending on the dataset
- Works for very large datasets
- Chances of being trapped in local minimum are lower



Cons

- Batch size is another hyper-parameter

Thank you