

# CM20315 - Machine Learning

Prof. Simon Prince  
6. Fitting models



# Regression

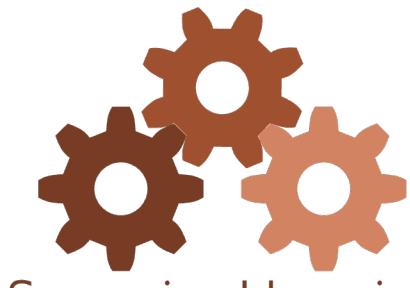
Real world input

6000 square feet,  
4 bedrooms,  
previously sold for  
\$235K in 2005,  
1 parking spot.

Model  
input

$$\begin{bmatrix} 6000 \\ 4 \\ 235 \\ 2005 \\ 1 \end{bmatrix}$$

Model



Supervised learning  
model

Model  
output

$$[340]$$

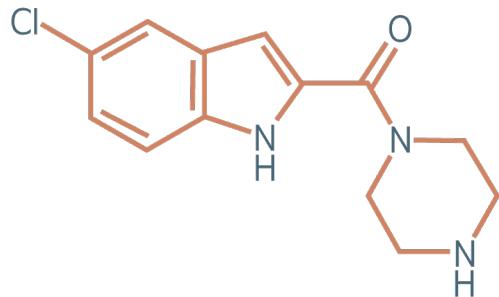
Real world output

Predicted price  
is \$340k

- Univariate regression problem (one output, real value)

# Graph regression

Real world input



Model  
input

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 17 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

Model



Supervised learning  
model

Model  
output

$$\begin{bmatrix} -12.9 \\ 56.4 \end{bmatrix}$$

Real world output

Freezing point  
is  $-12.9^{\circ}\text{C}$   
Boiling point  
is  $56.4^{\circ}\text{C}$

- Multivariate regression problem (>1 output, real value)

# Text classification

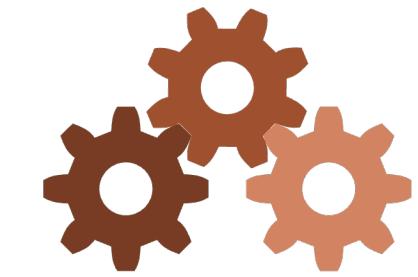
Real world input

"The steak was terrible,  
the salad was rotten, and  
the soup tasted like socks"

Model  
input

$$\begin{bmatrix} 8672 \\ 8194 \\ 9804 \\ 8634 \\ 8672 \\ \vdots \end{bmatrix}$$

Model



Supervised learning  
model

Model  
output

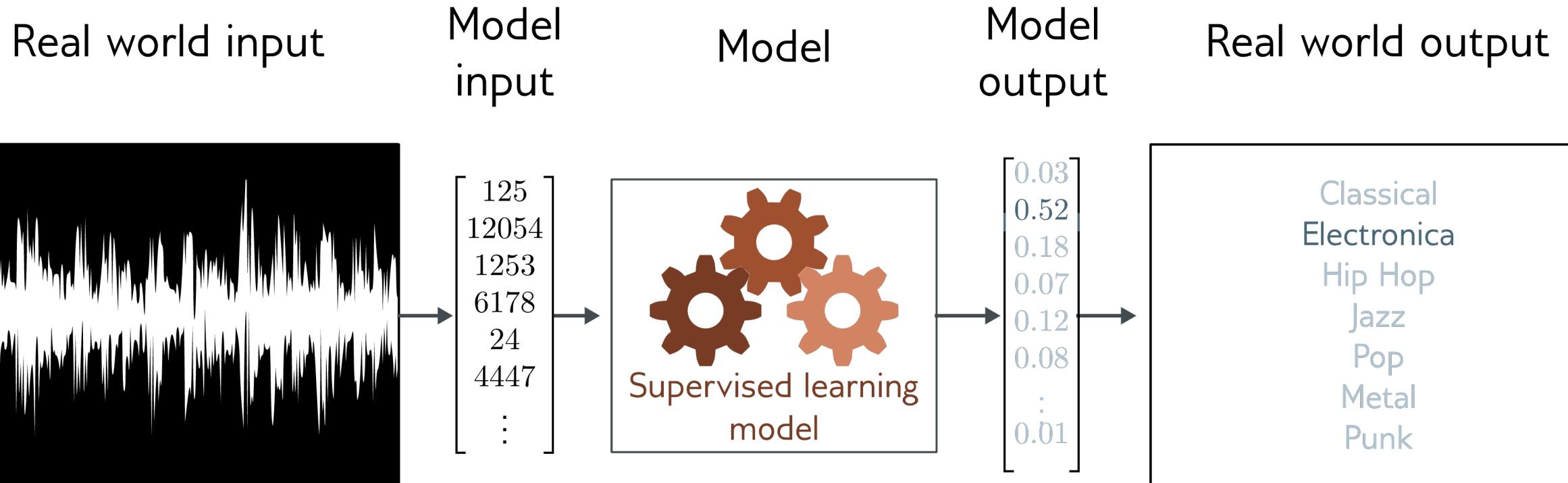
$$\begin{bmatrix} 0.02 \\ 0.98 \end{bmatrix}$$

Real world output

Positive  
Negative

- Binary classification problem (two discrete classes)

# Music genre classification



- Multiclass classification problem (discrete classes, >2 possible values)

# 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:

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

or for short:

$$L[\phi]$$

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

# Training

- Loss function:

$$L[\phi]$$

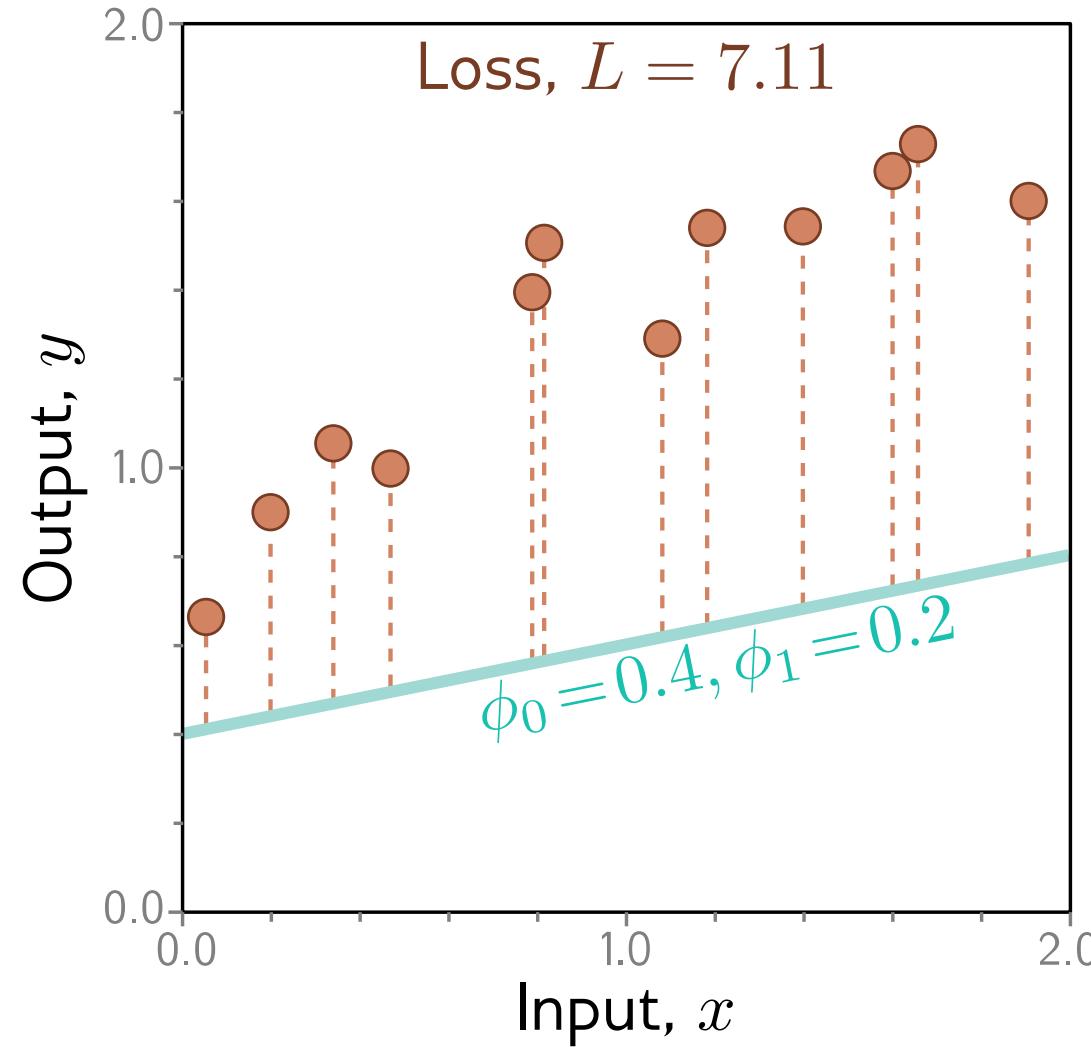


Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

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

# Example: 1D Linear regression loss function

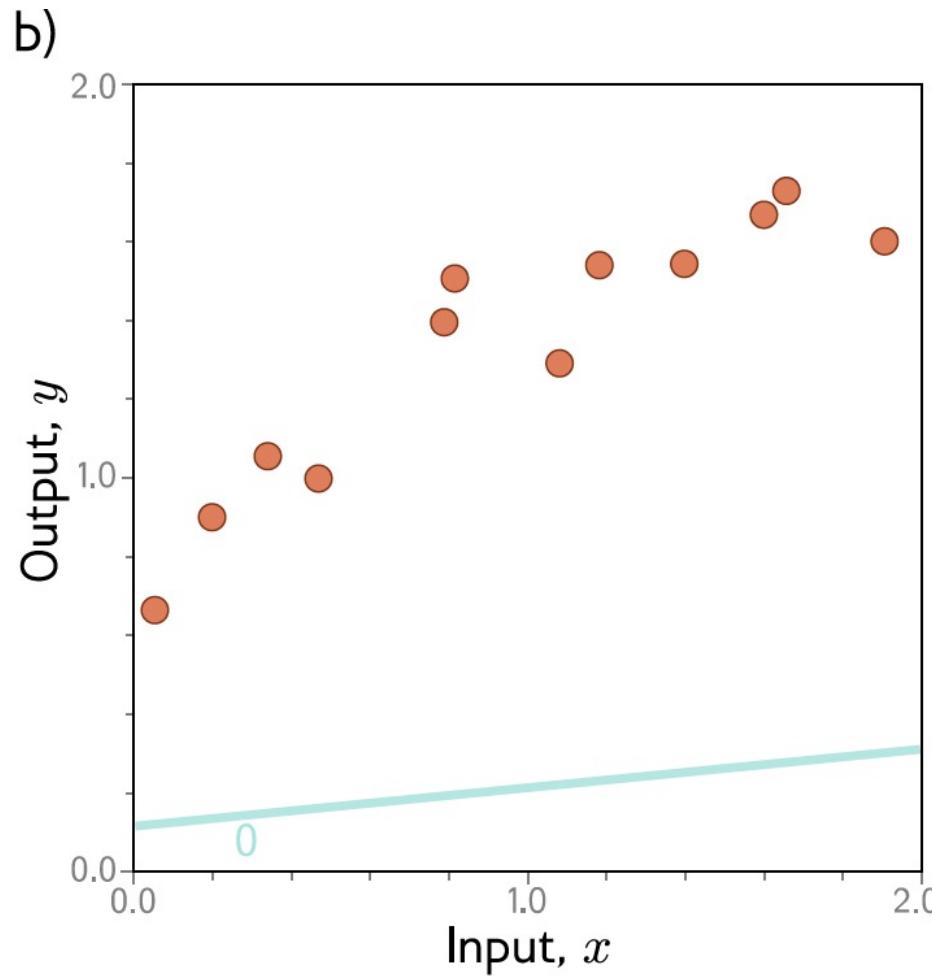
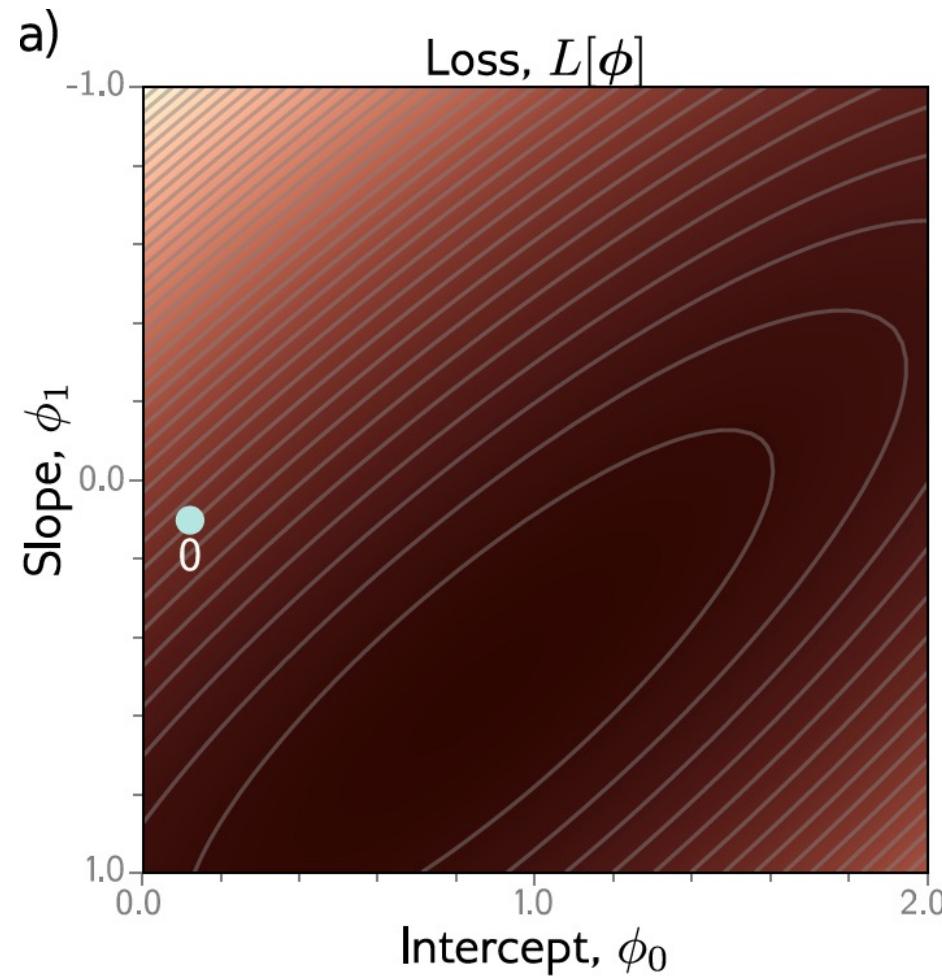


Loss function:

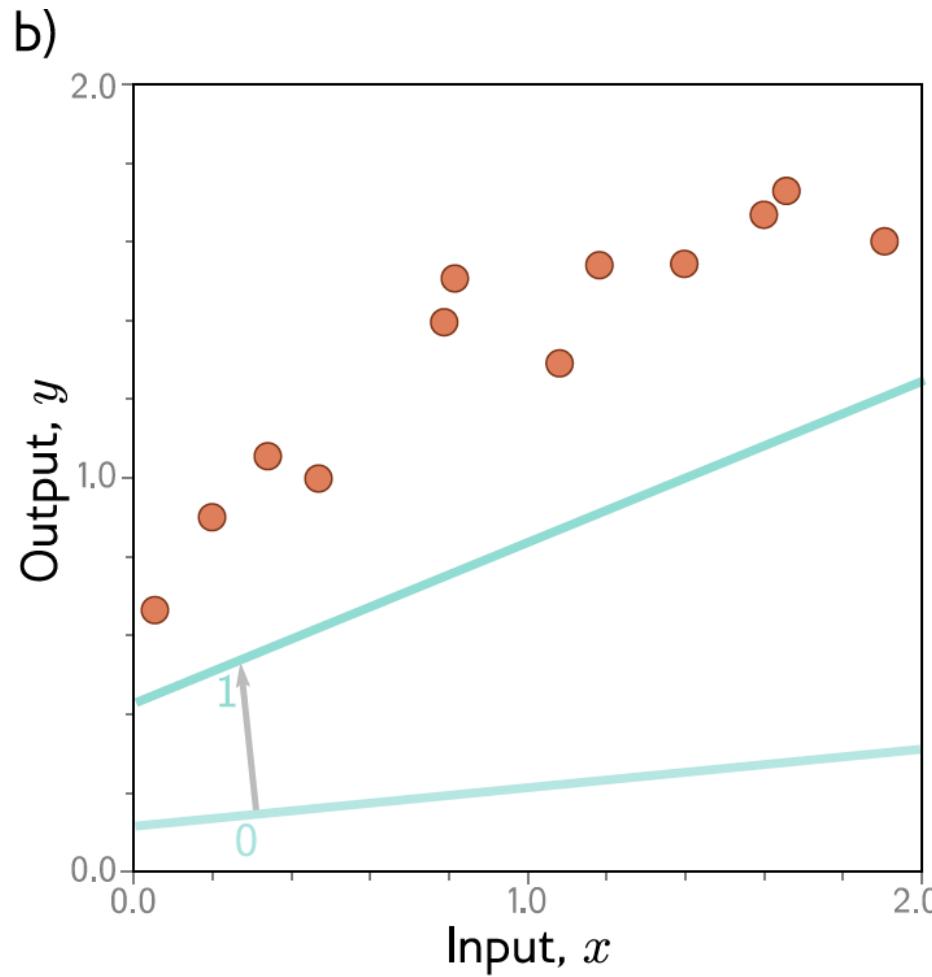
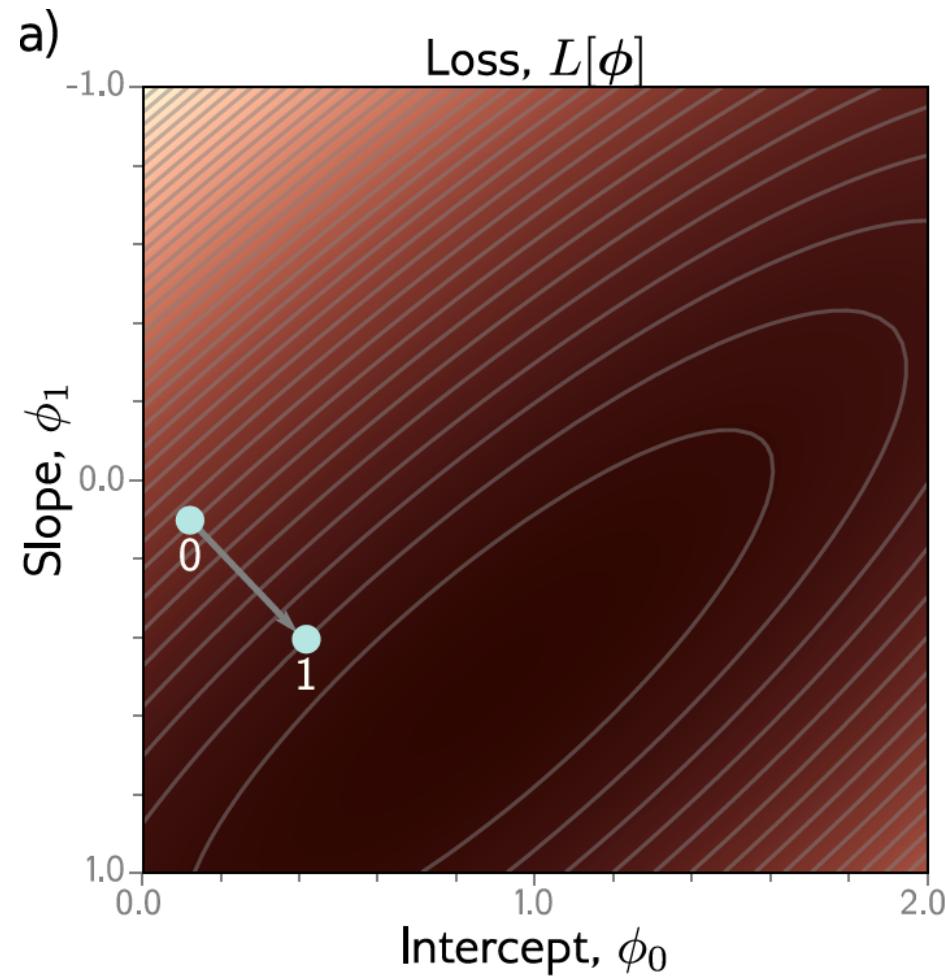
$$\begin{aligned} L[\phi] &= \sum_{i=1}^I (f[x_i, \phi] - y_i)^2 \\ &= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2 \end{aligned}$$

“Least squares loss function”

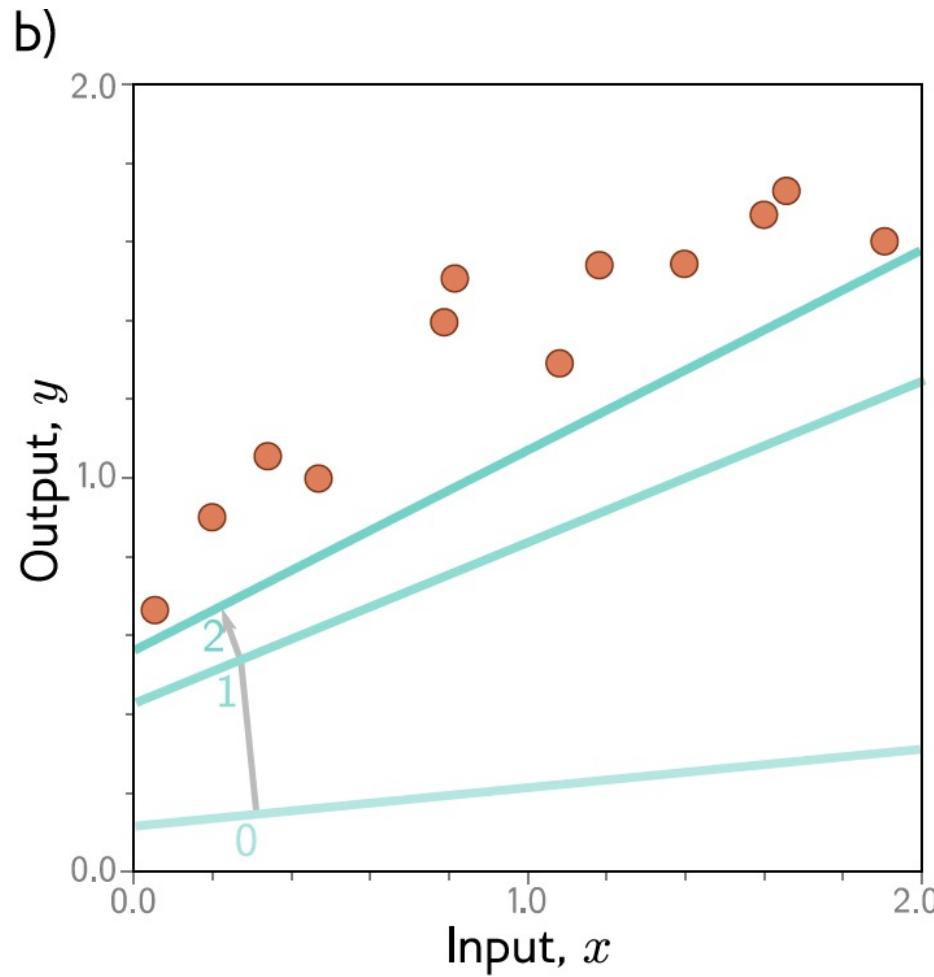
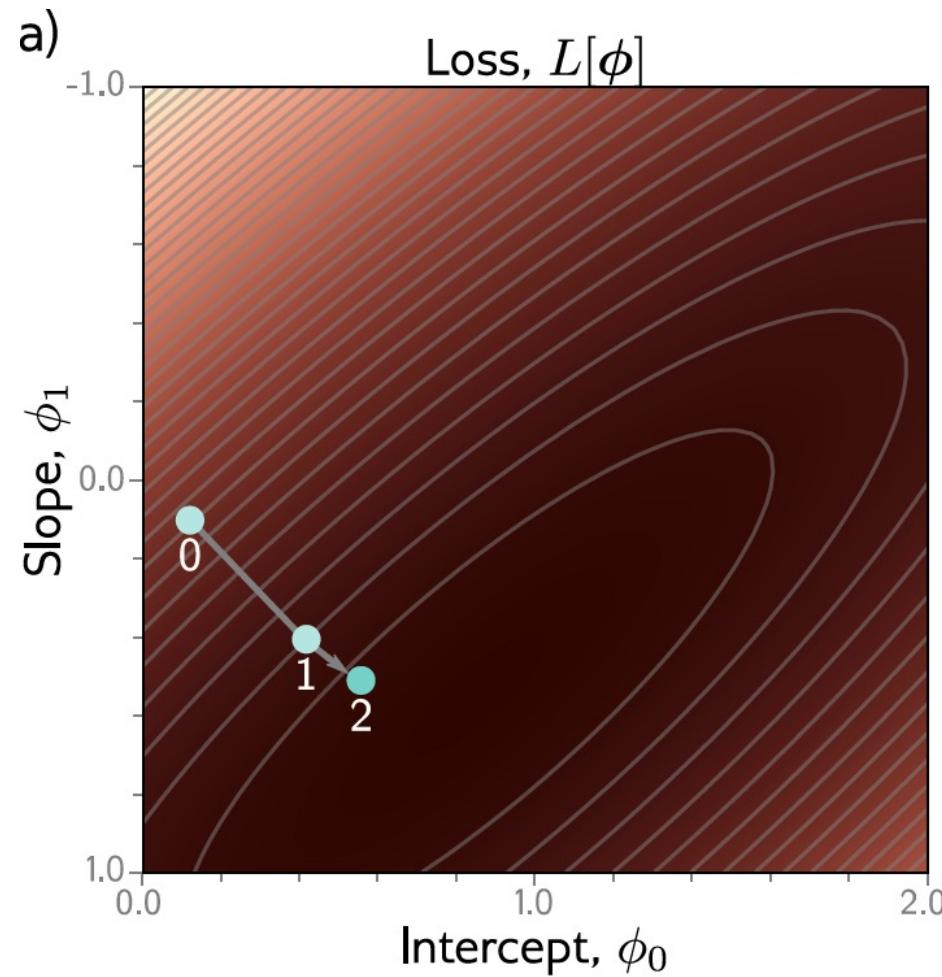
# Example: 1D Linear regression training



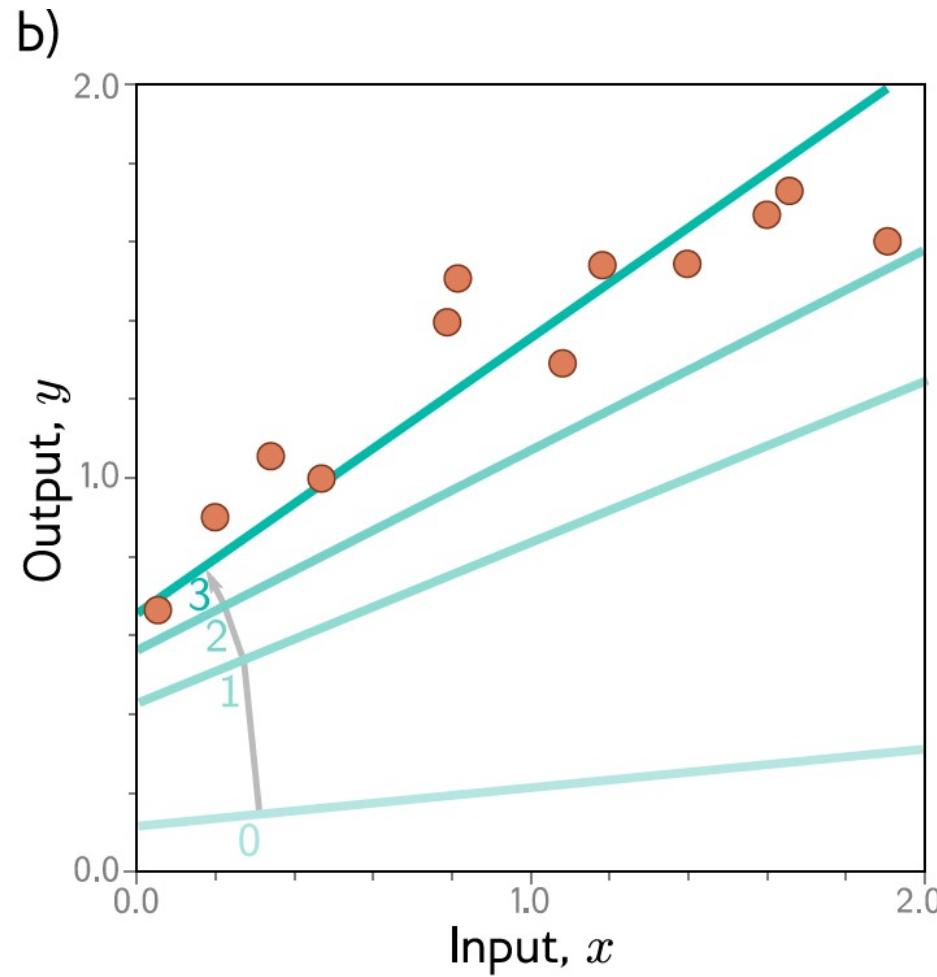
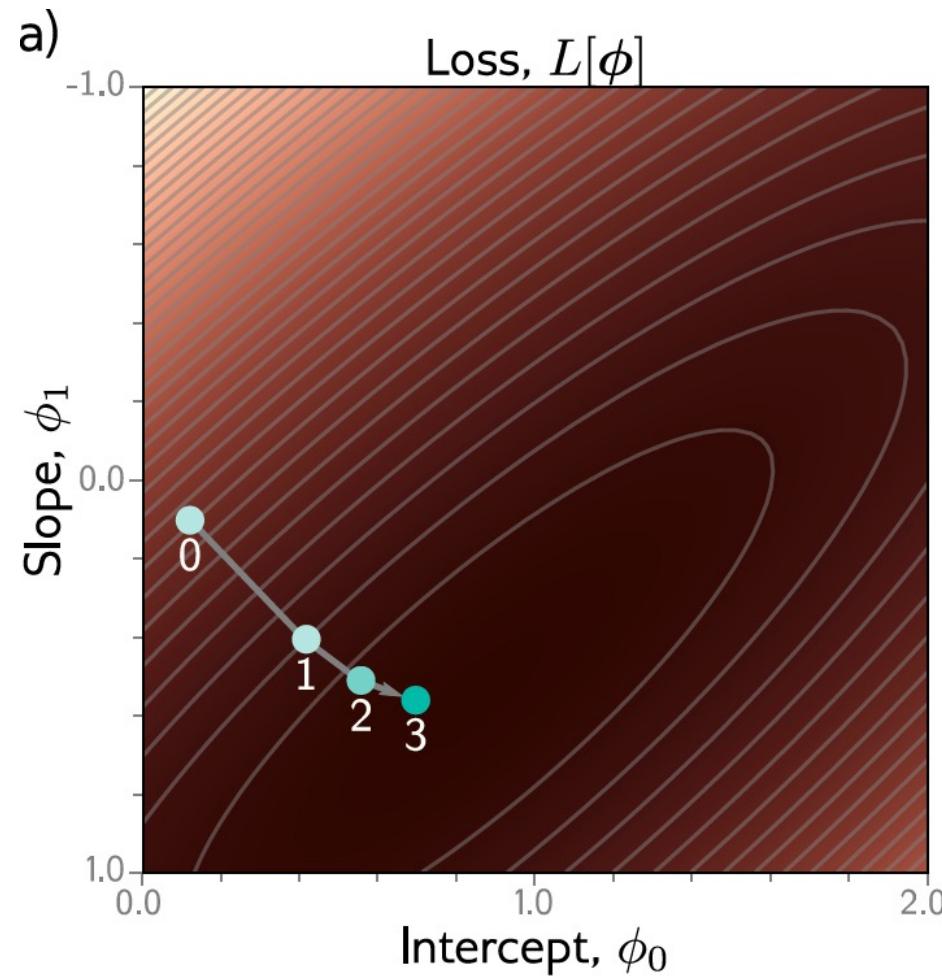
# Example: 1D Linear regression training



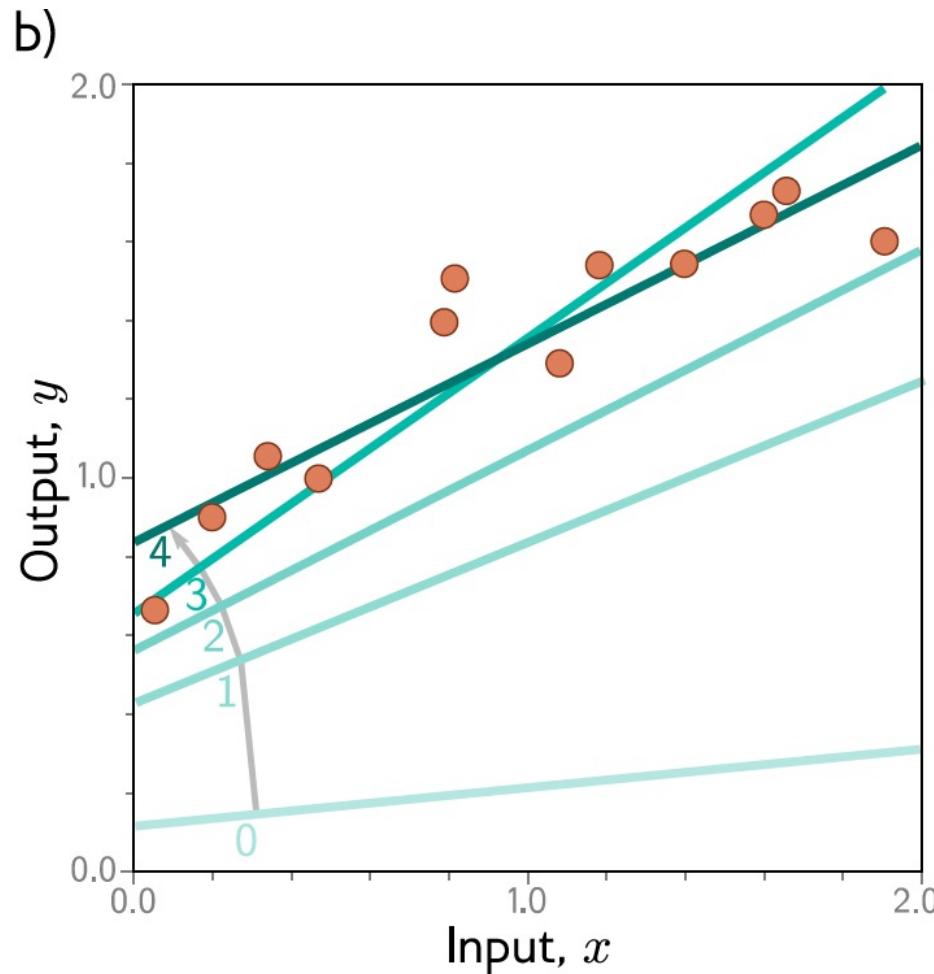
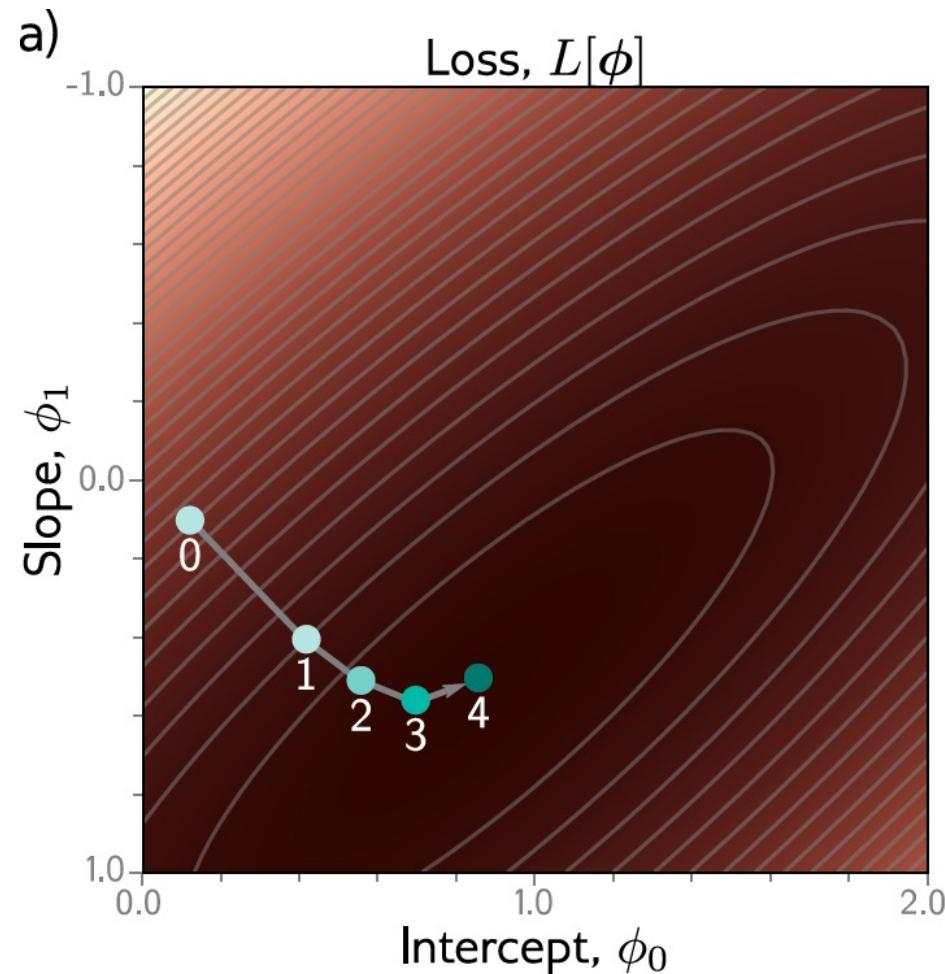
# Example: 1D Linear regression training



# Example: 1D Linear regression training



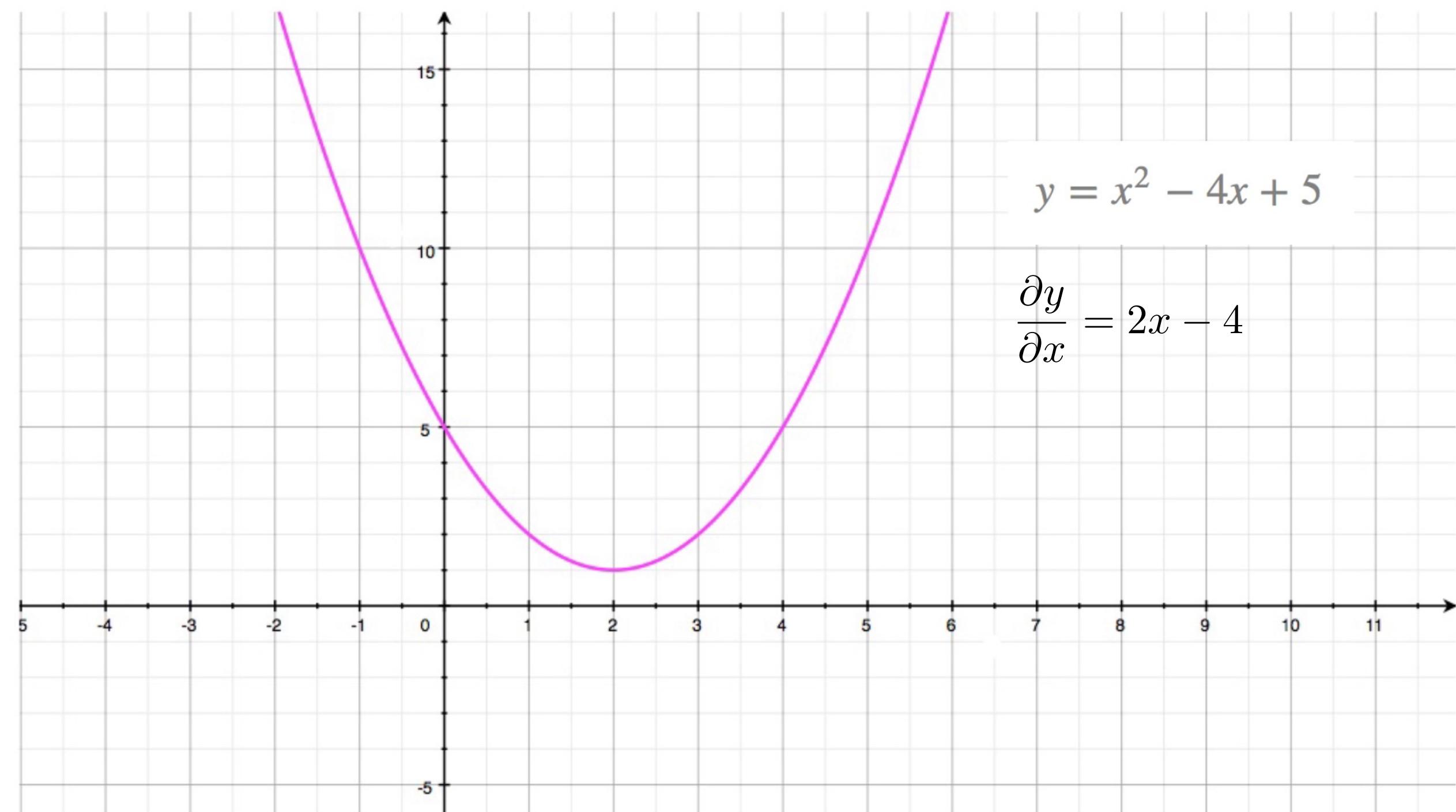
# Example: 1D Linear regression training



This technique is known as **gradient descent**

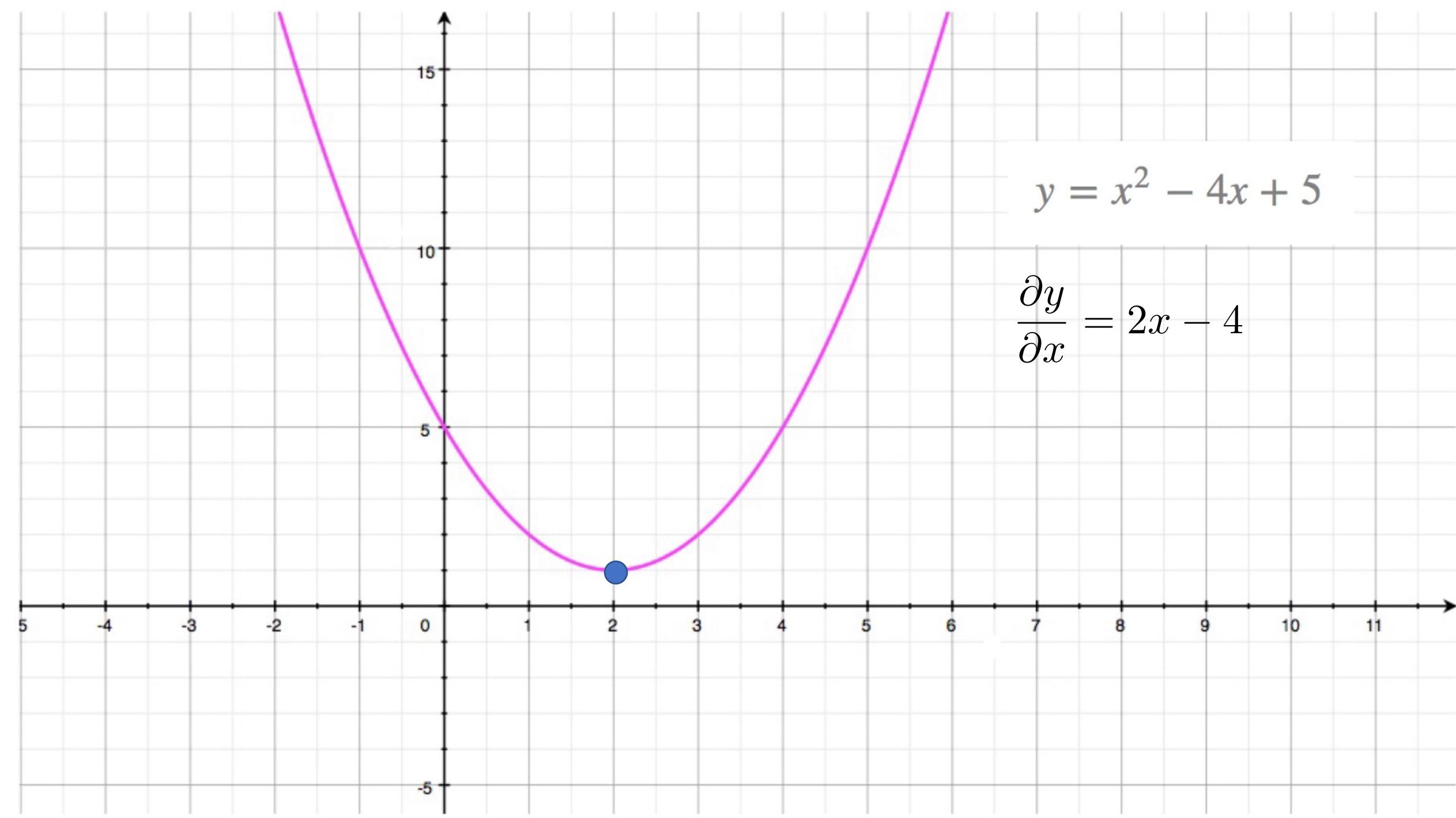
# Fitting models

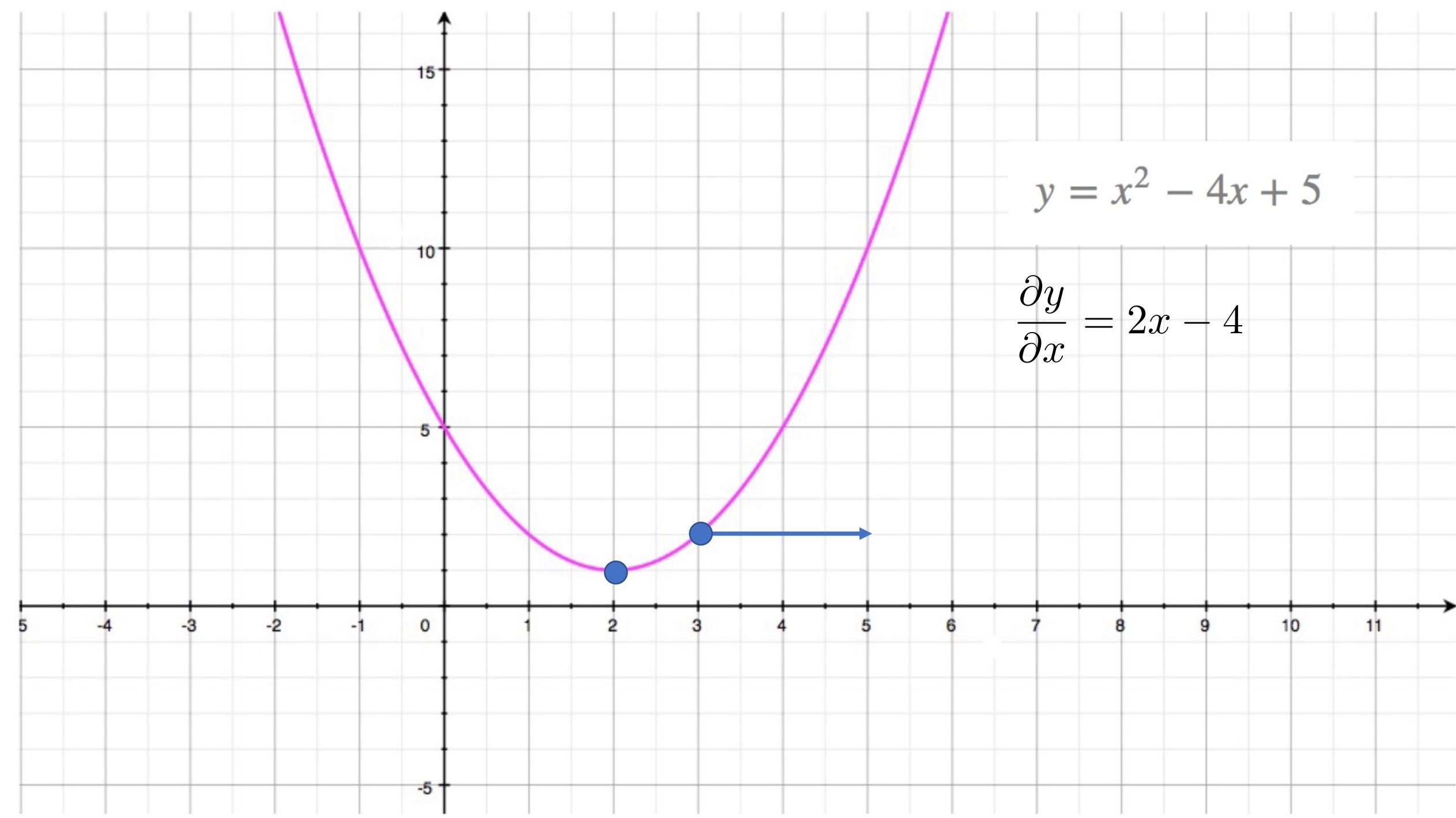
- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam



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

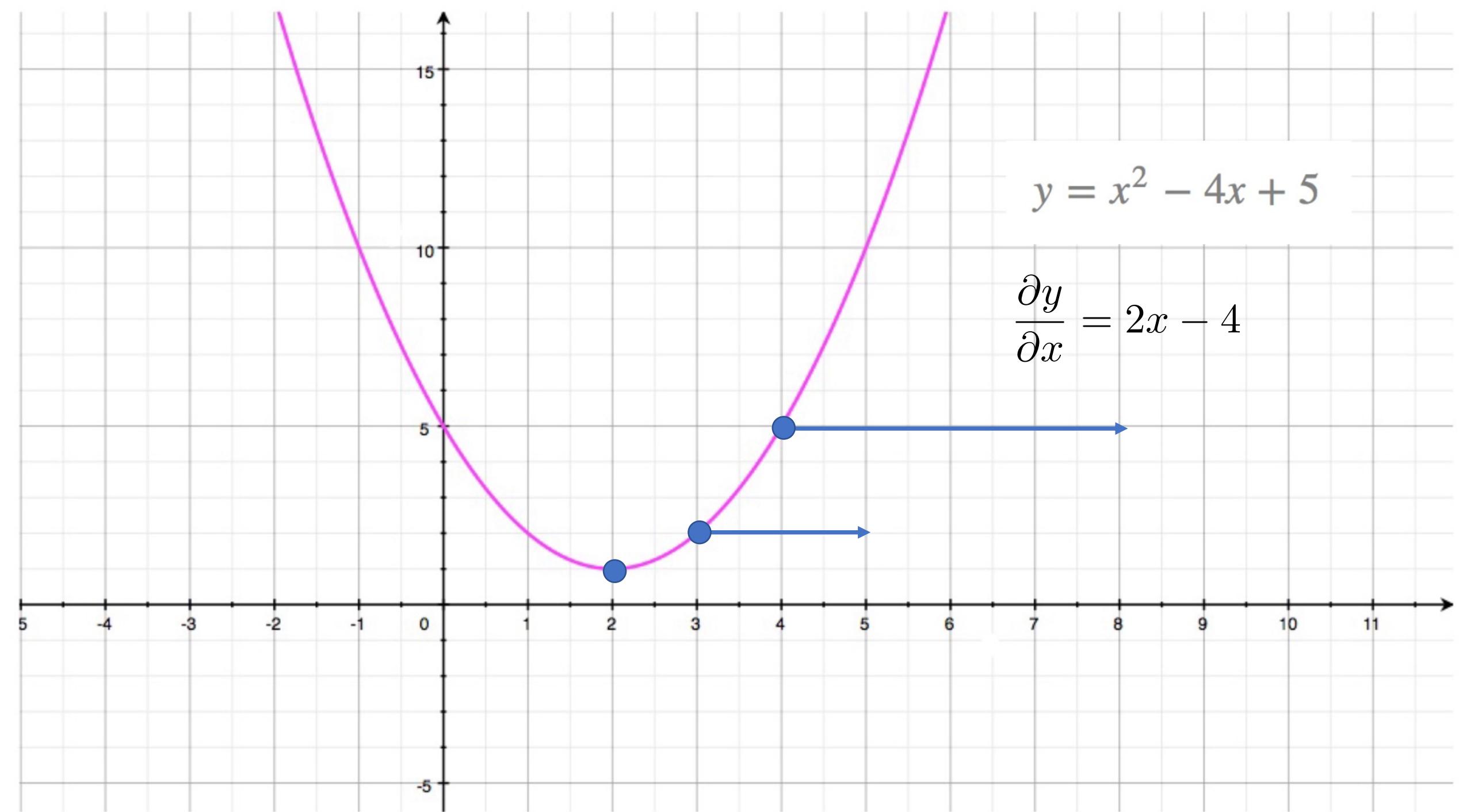
$$\frac{\partial y}{\partial x} = 2x - 4$$

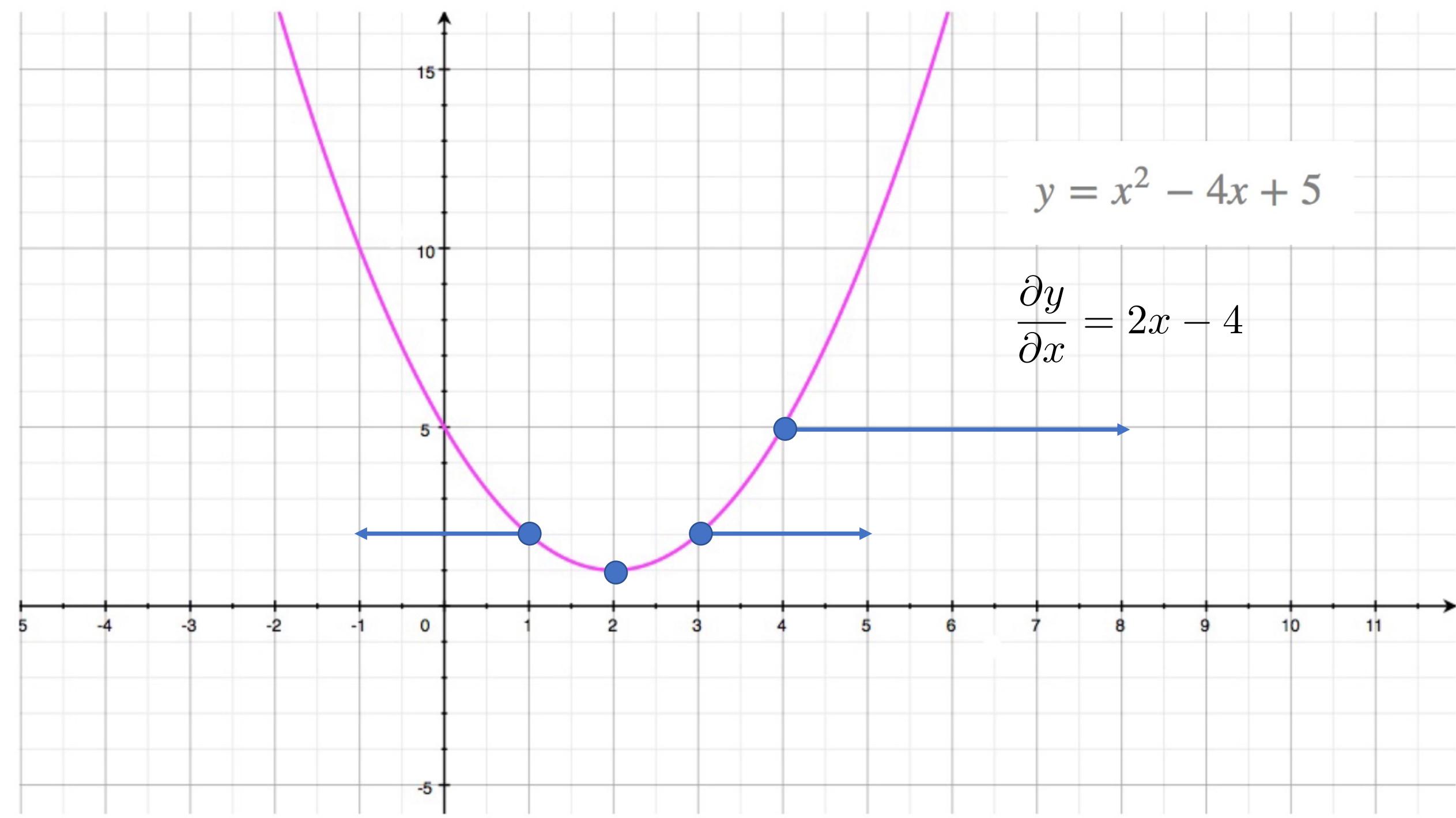




$$y = x^2 - 4x + 5$$

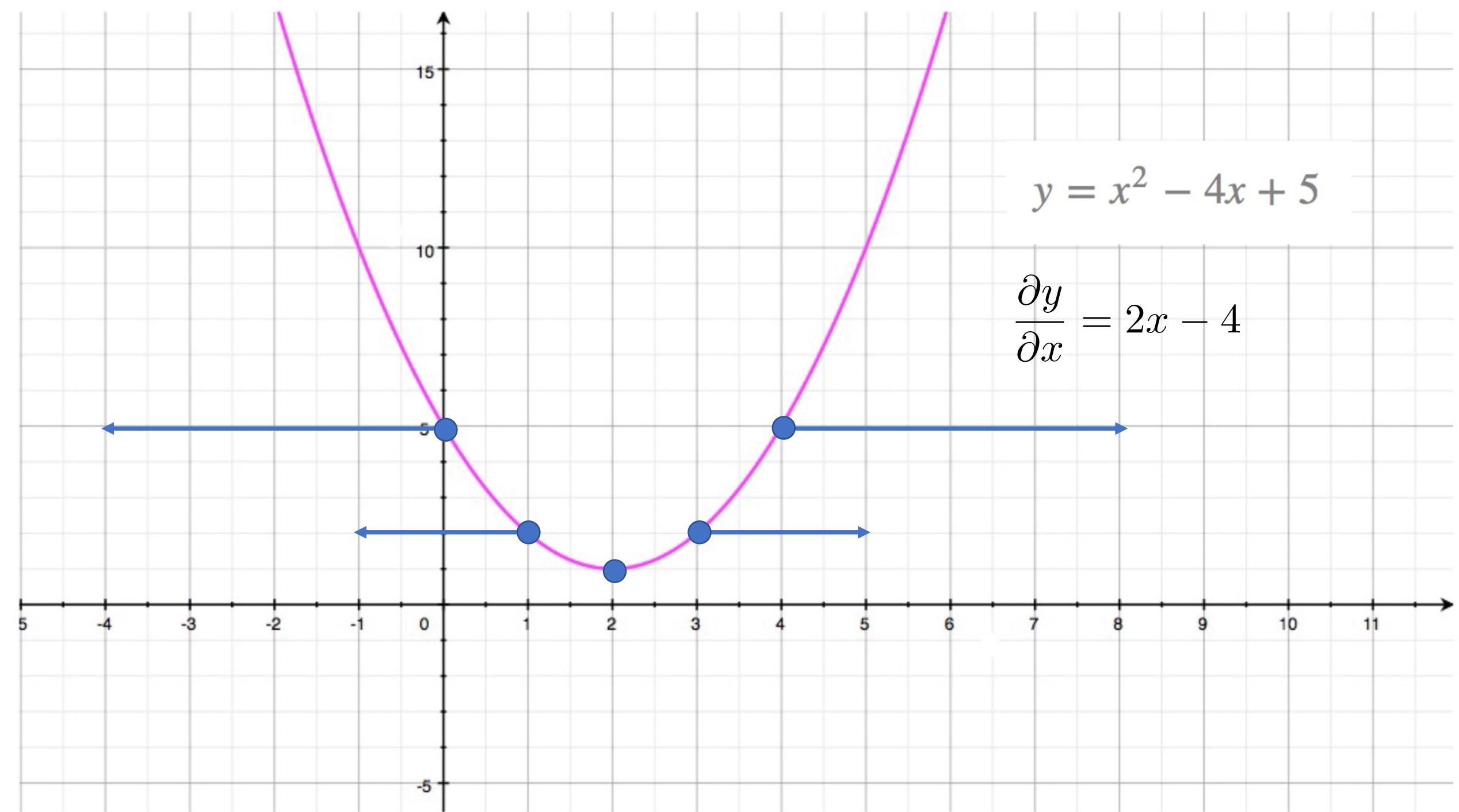
$$\frac{\partial y}{\partial x} = 2x - 4$$





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

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



# Fitting models

- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

# Gradient descent algorithm

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

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

**Step 2.** Update the parameters according to the rule:

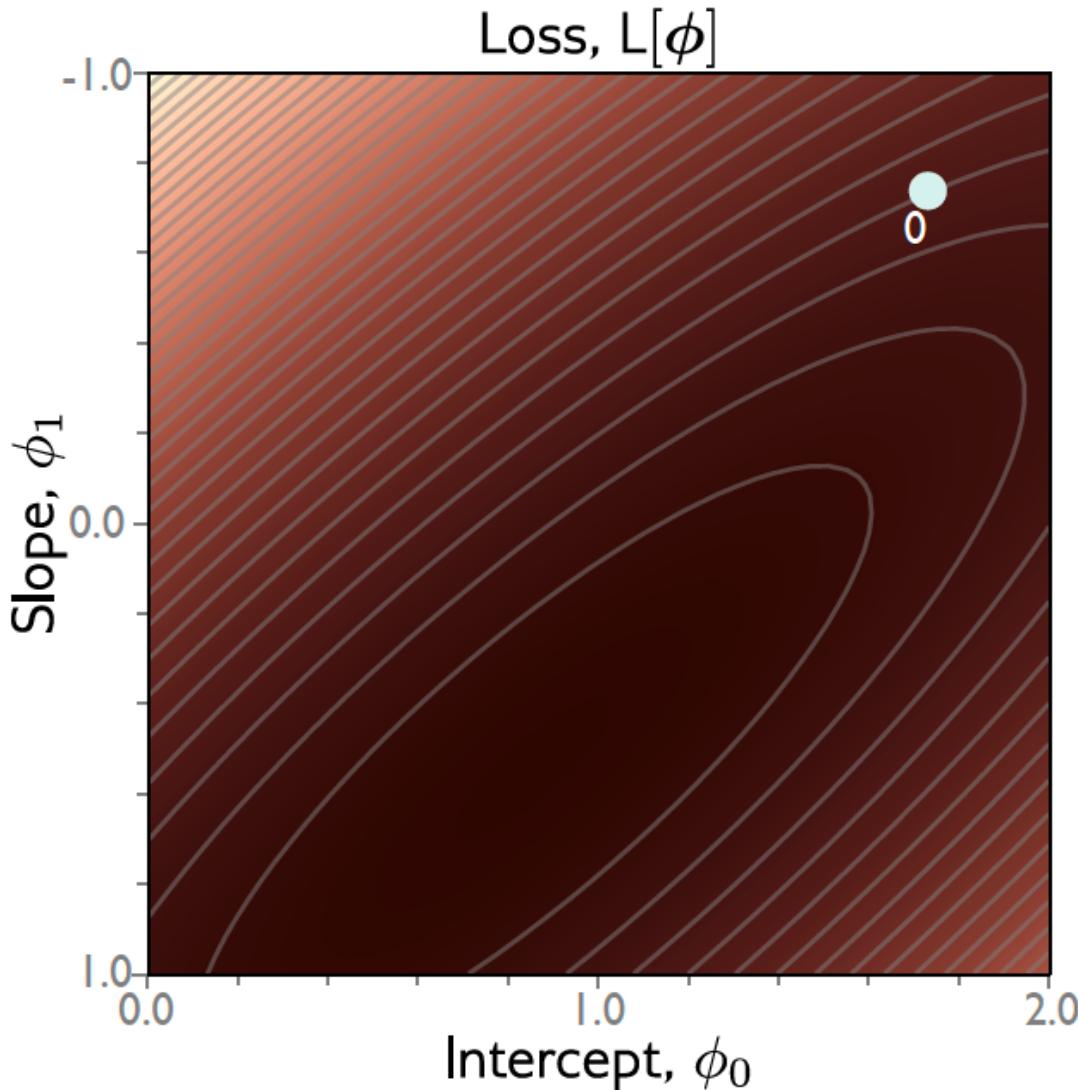
$$\phi \leftarrow \phi - \alpha \frac{\partial L}{\partial \phi},$$

where the positive scalar  $\alpha$  determines the magnitude of the change.

# Fitting models

- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

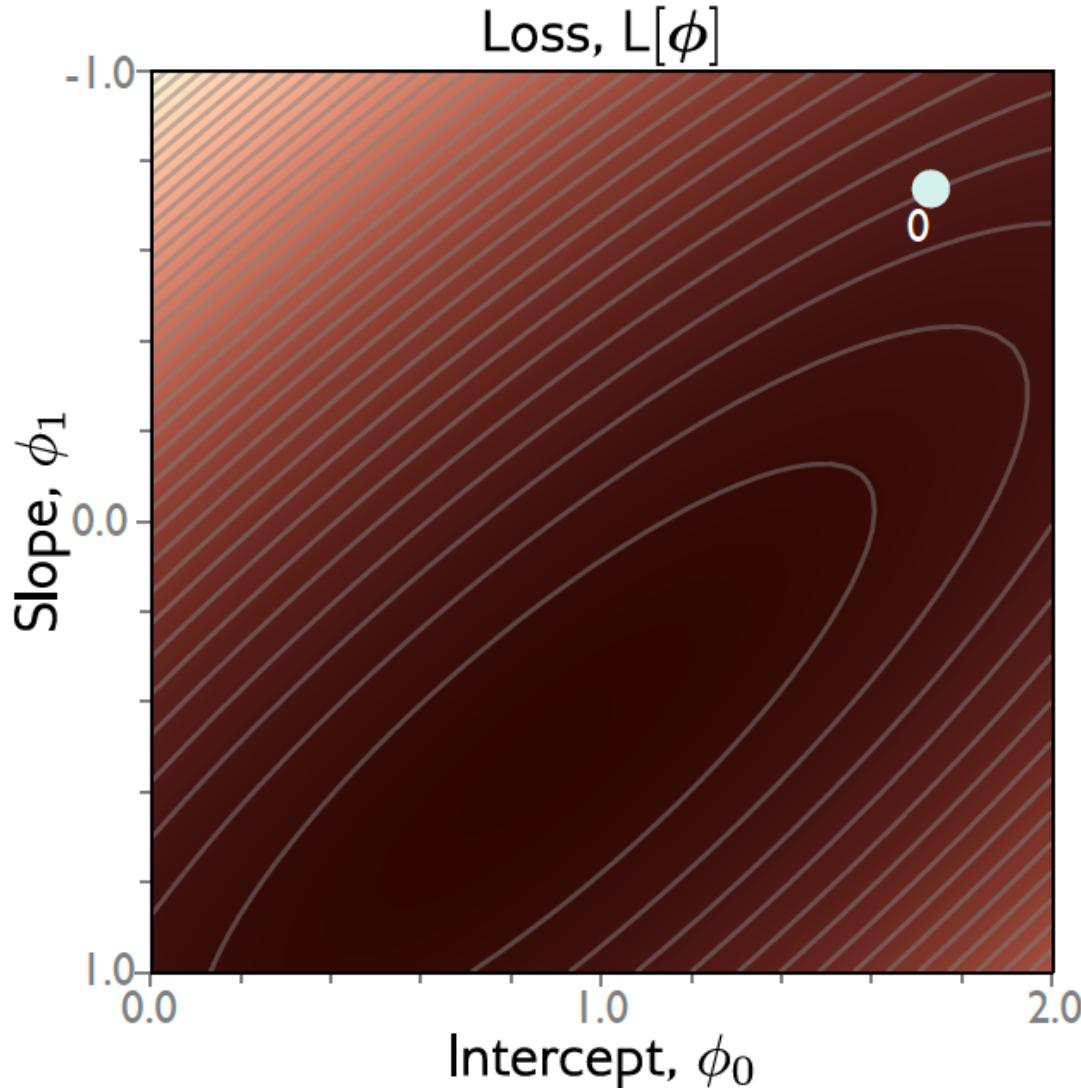
# Gradient descent



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

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

# Gradient descent

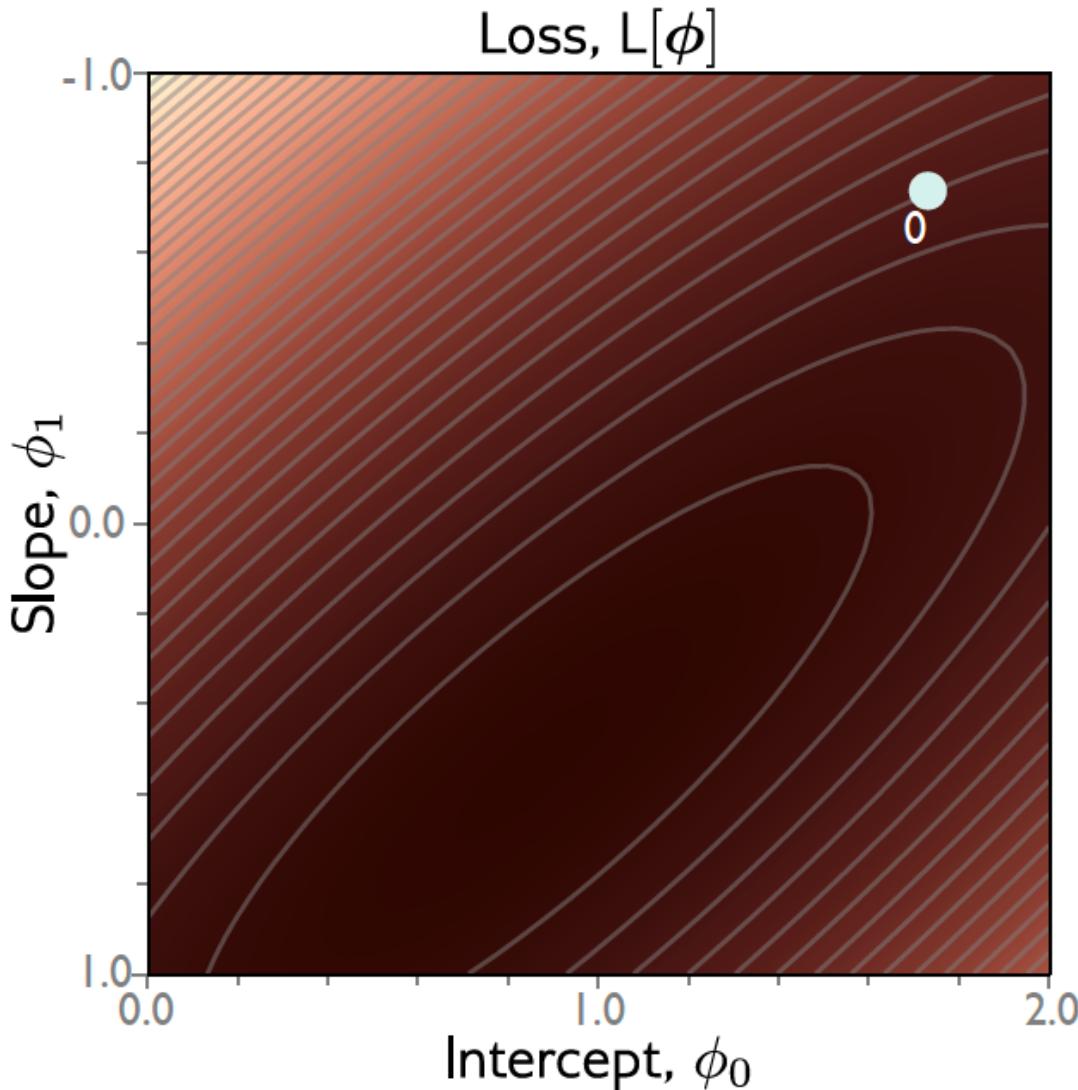


Step 1: Compute derivatives (slopes of function) with Respect to the parameters

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

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

# Gradient descent



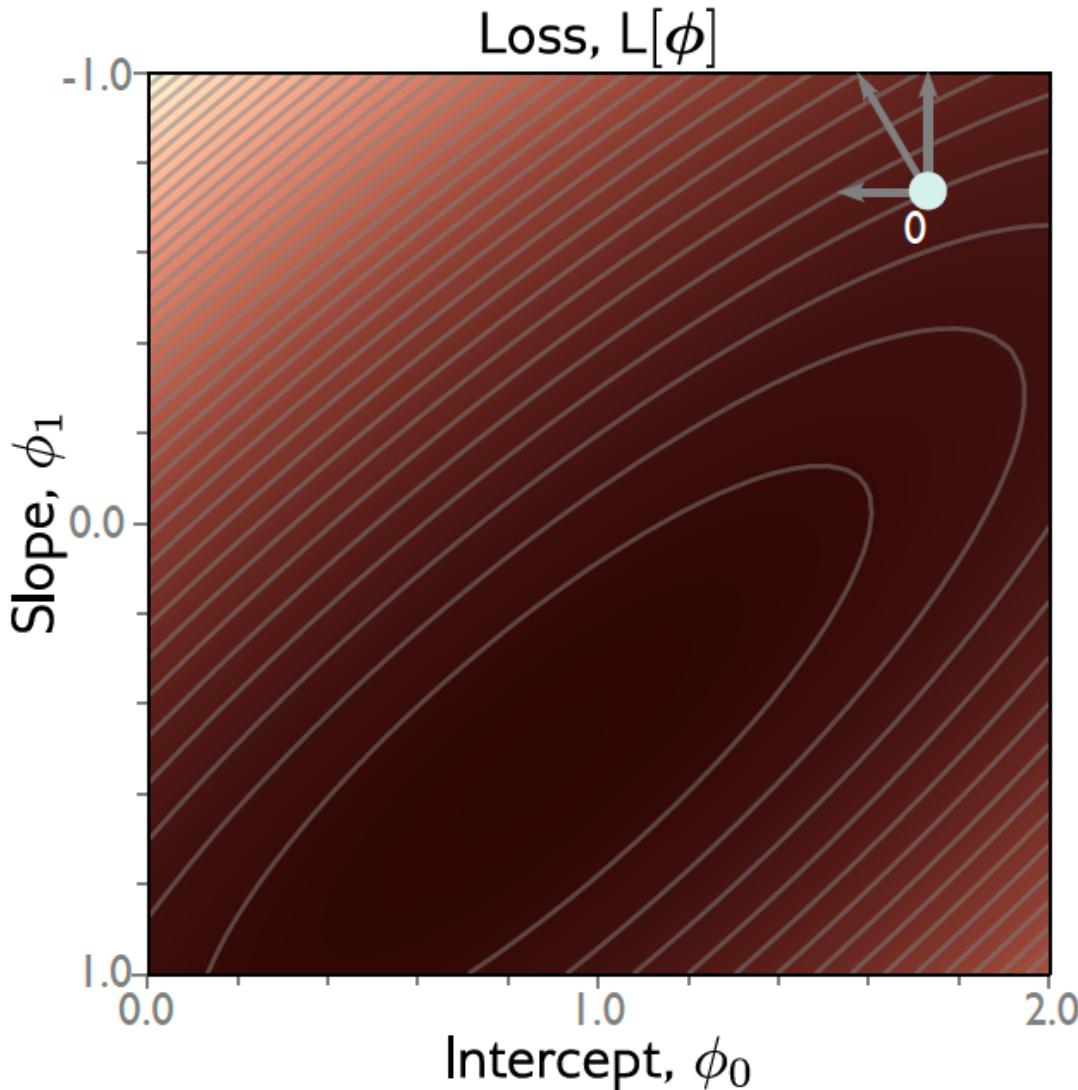
Step 1: Compute derivatives (slopes of function) with Respect to the parameters

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

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

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

# Gradient descent

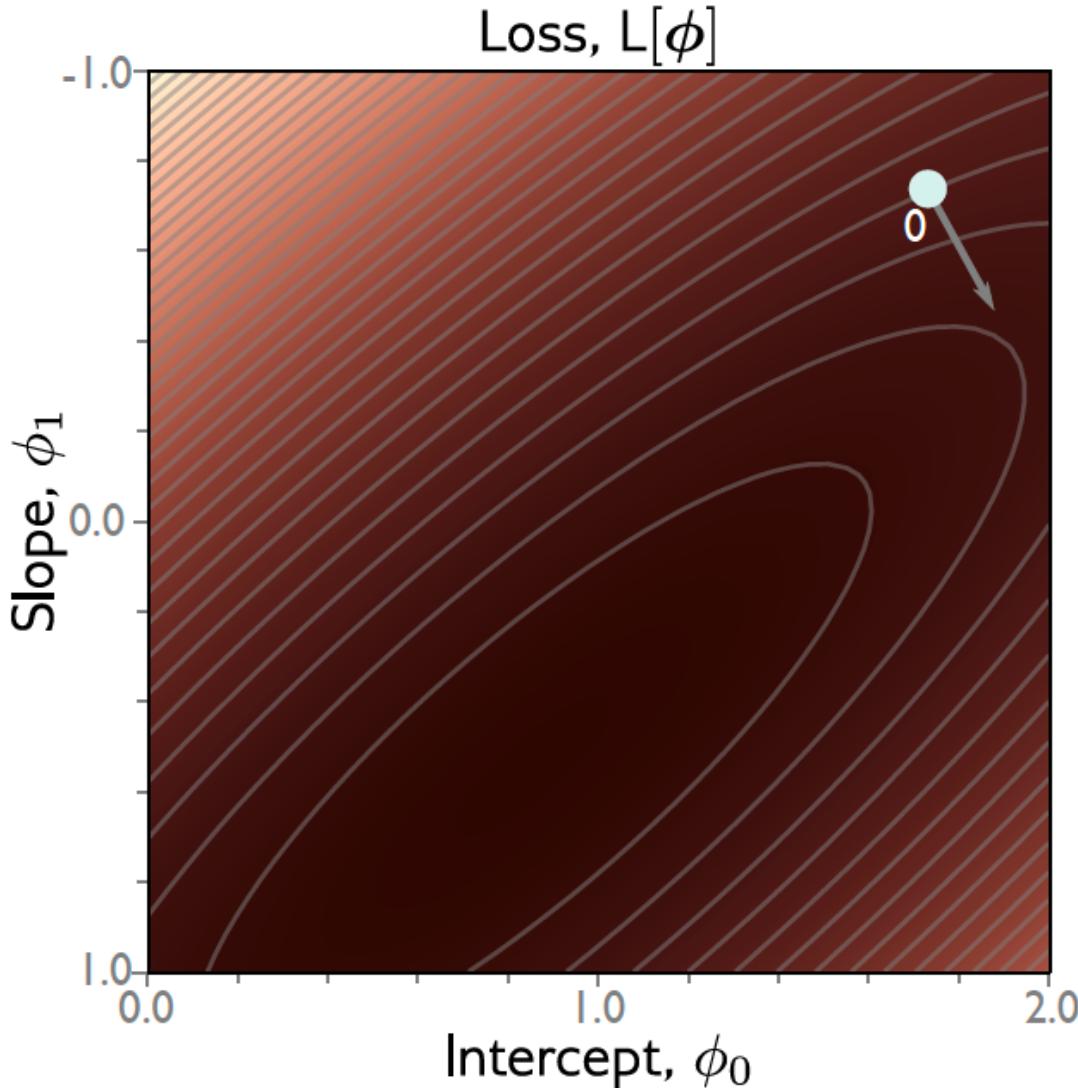


Step 1: Compute derivatives (slopes of function) with Respect to the parameters

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

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

# Gradient descent



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

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

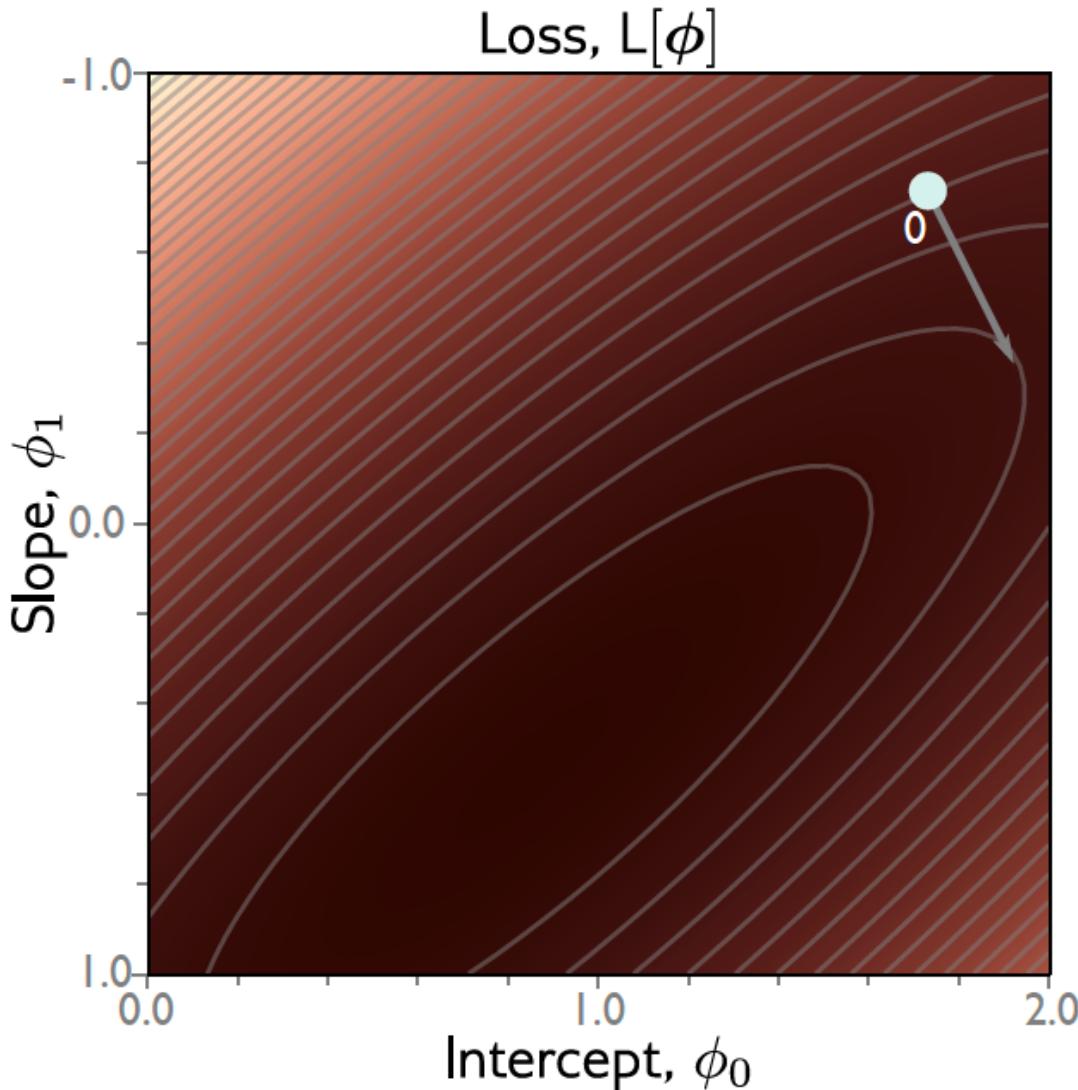
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

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

$\alpha$  = step size or learning rate if fixed

# Gradient descent



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

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

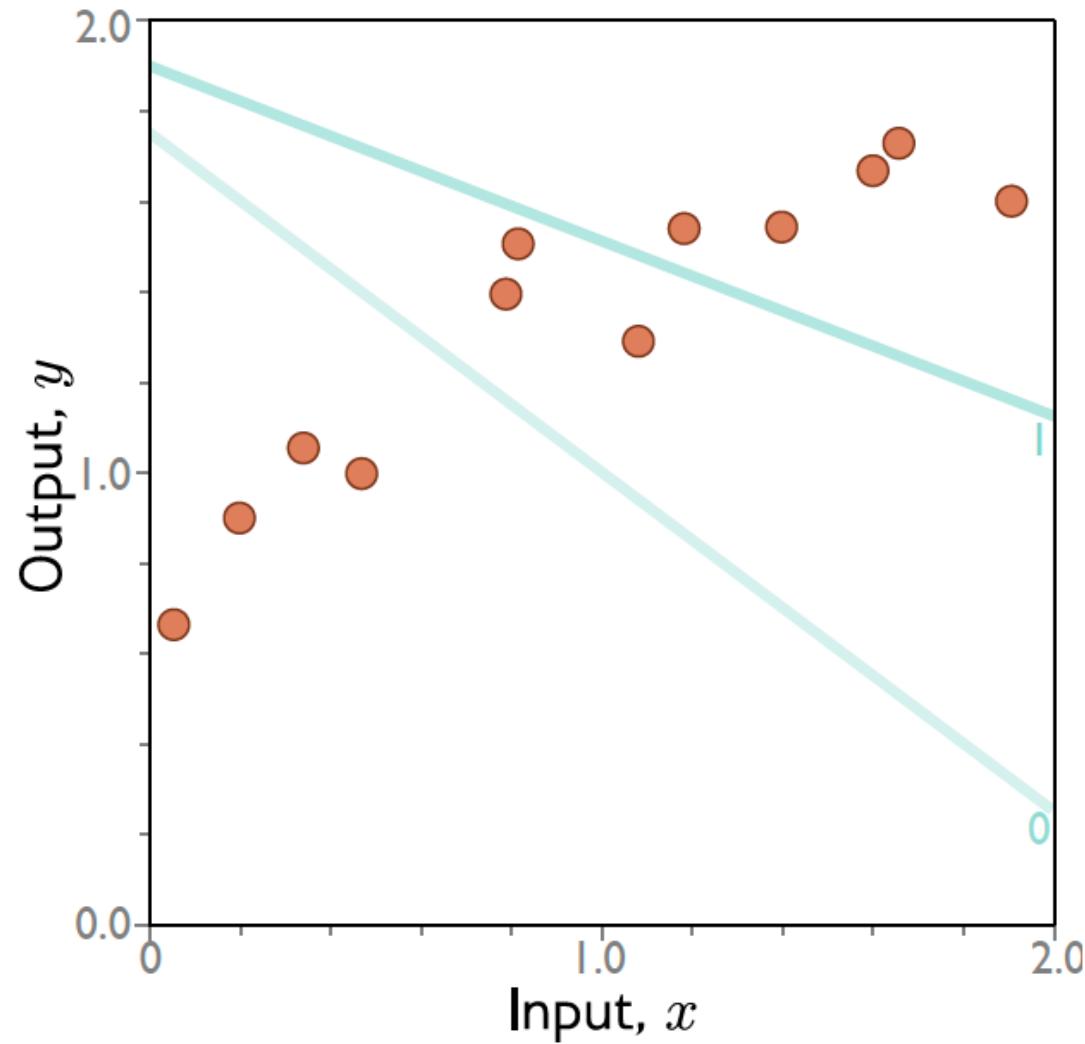
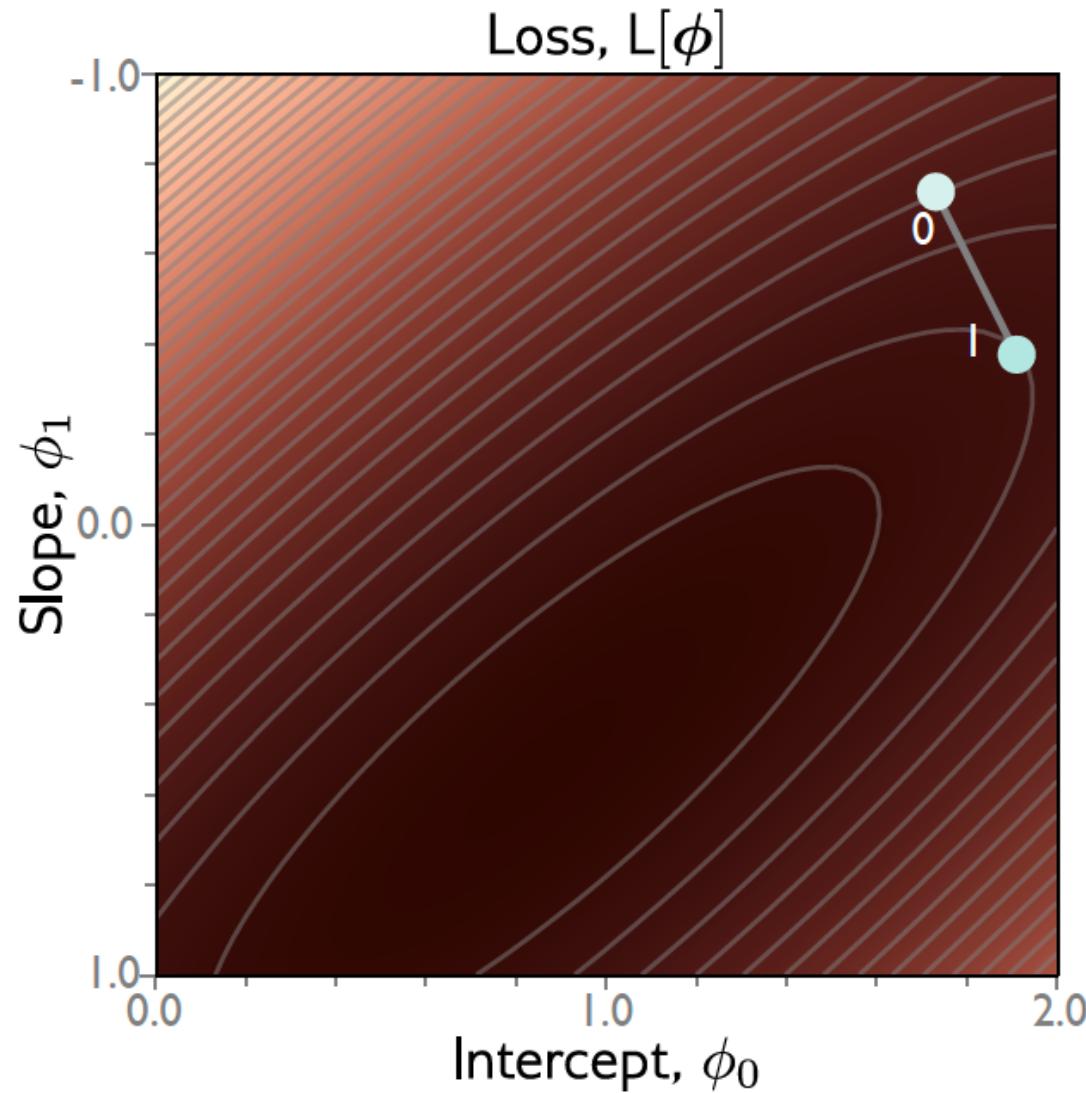
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

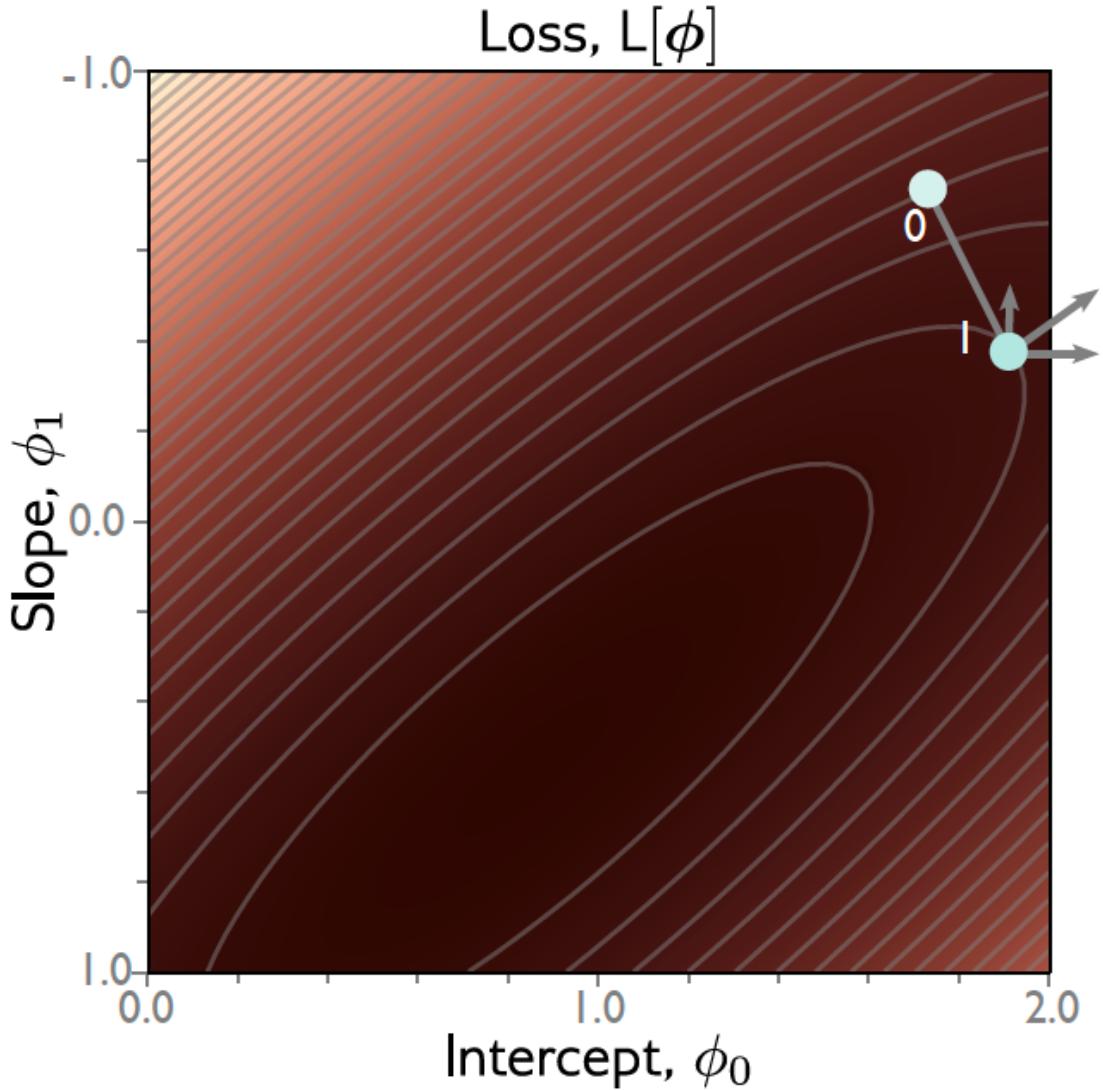
$$\phi \leftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

$\alpha$  = step size

# Gradient descent



# Gradient descent



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

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

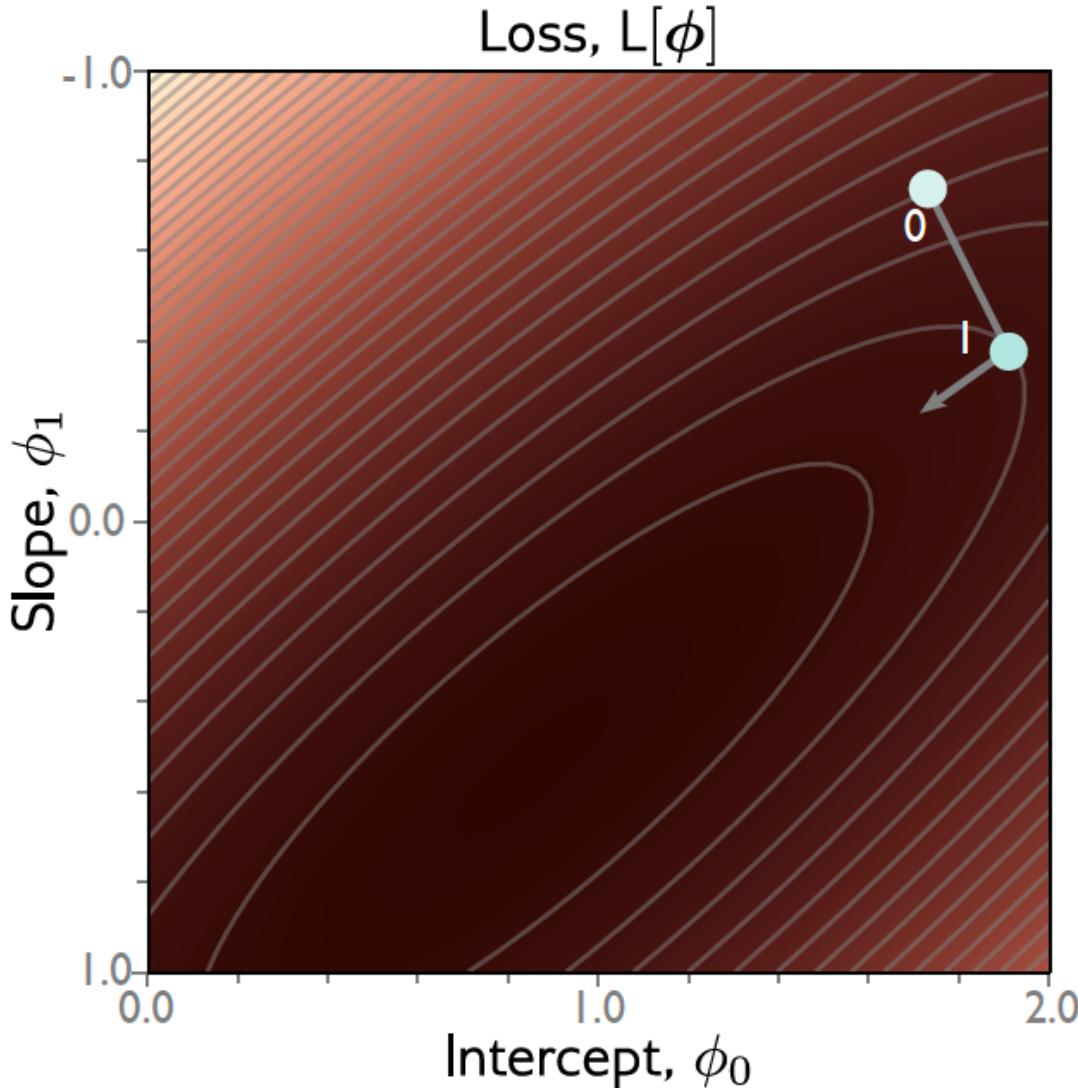
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

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

$\alpha$  = step size

# Gradient descent



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

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

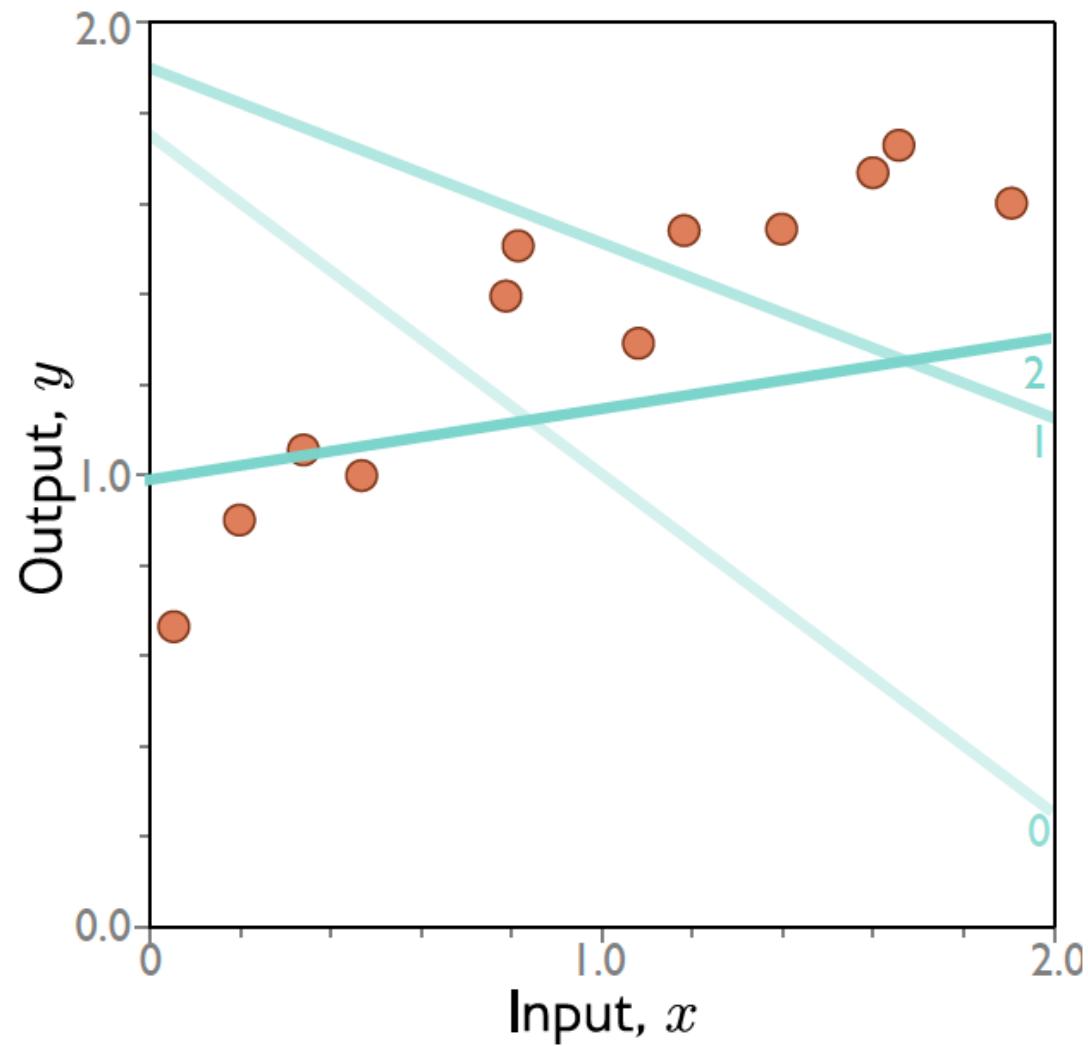
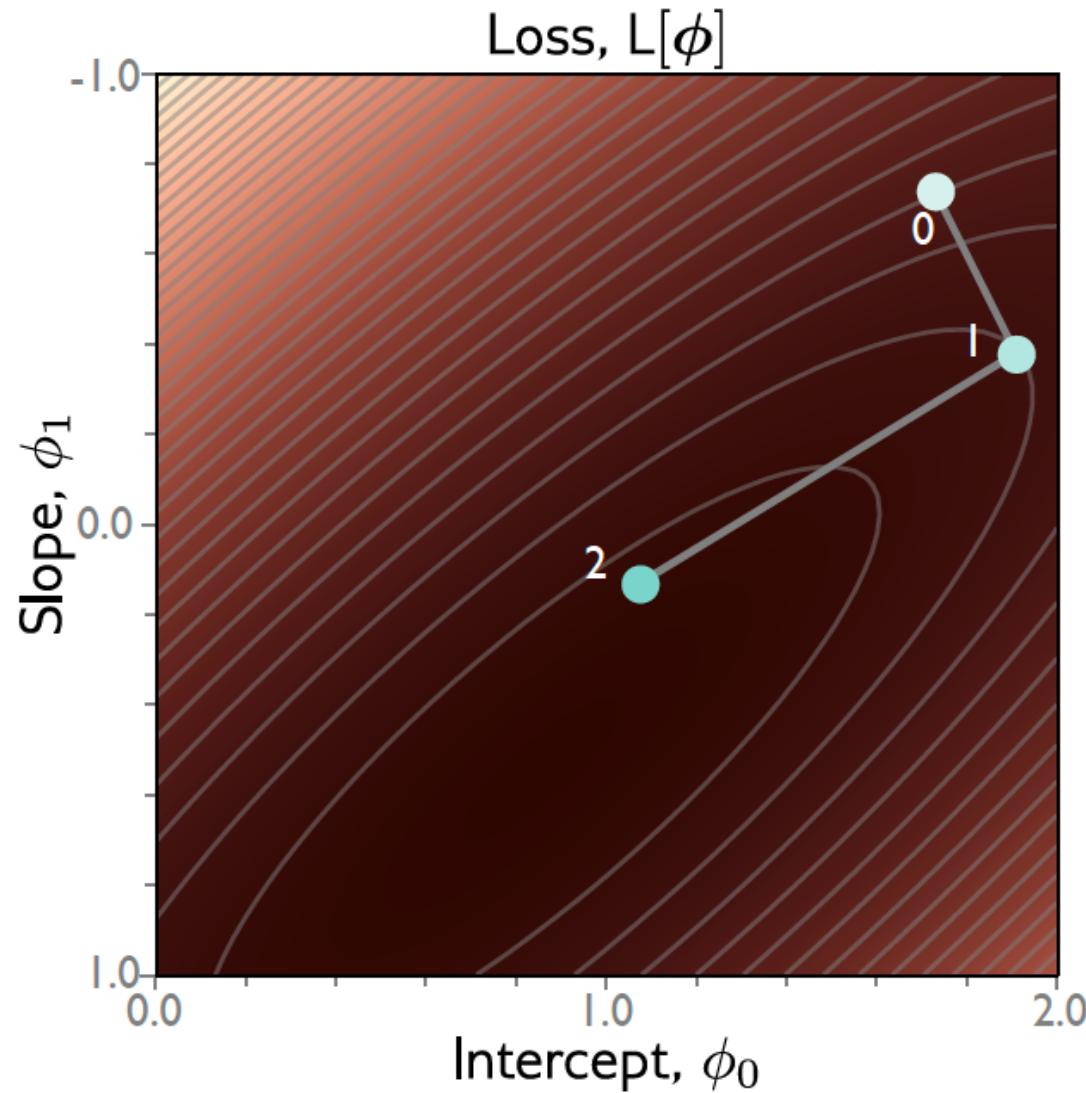
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

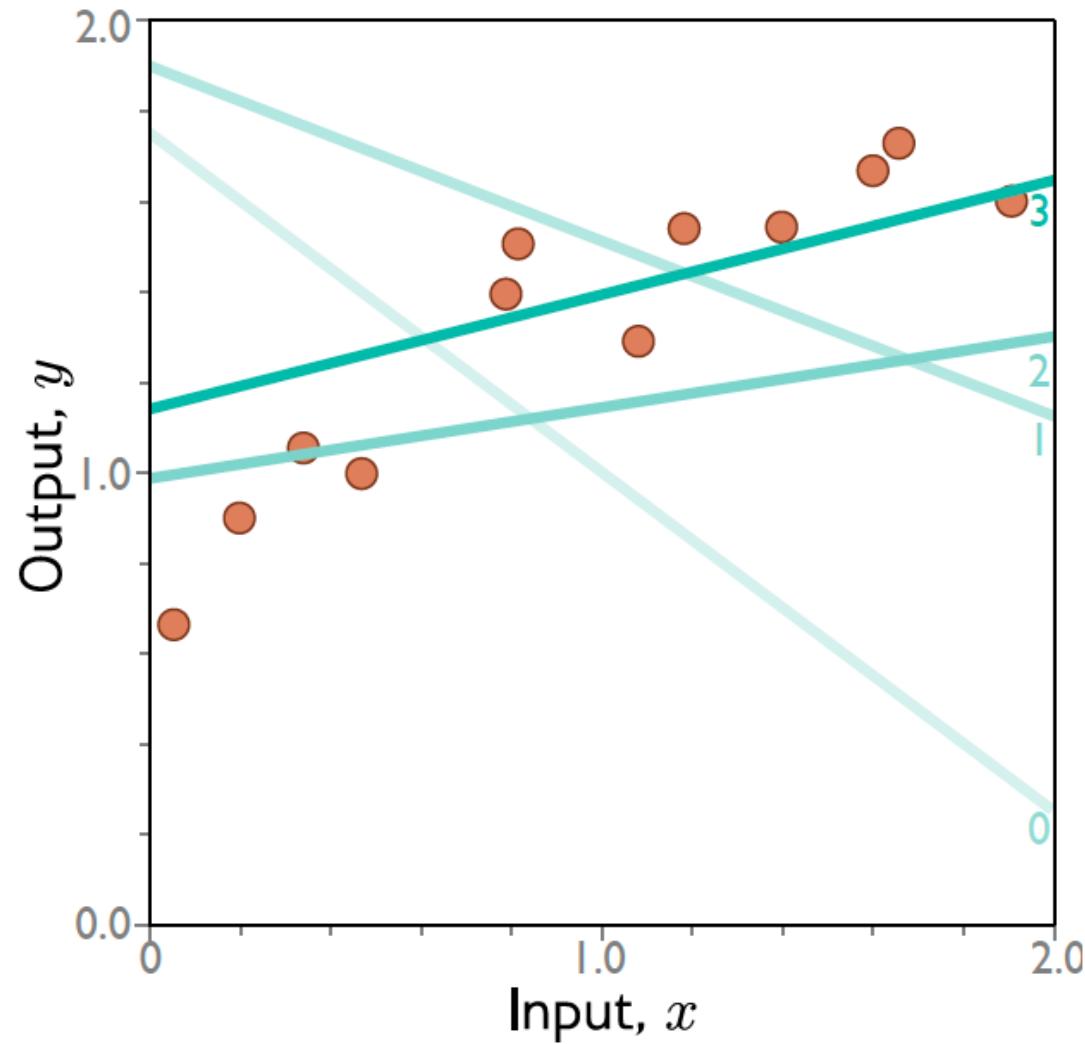
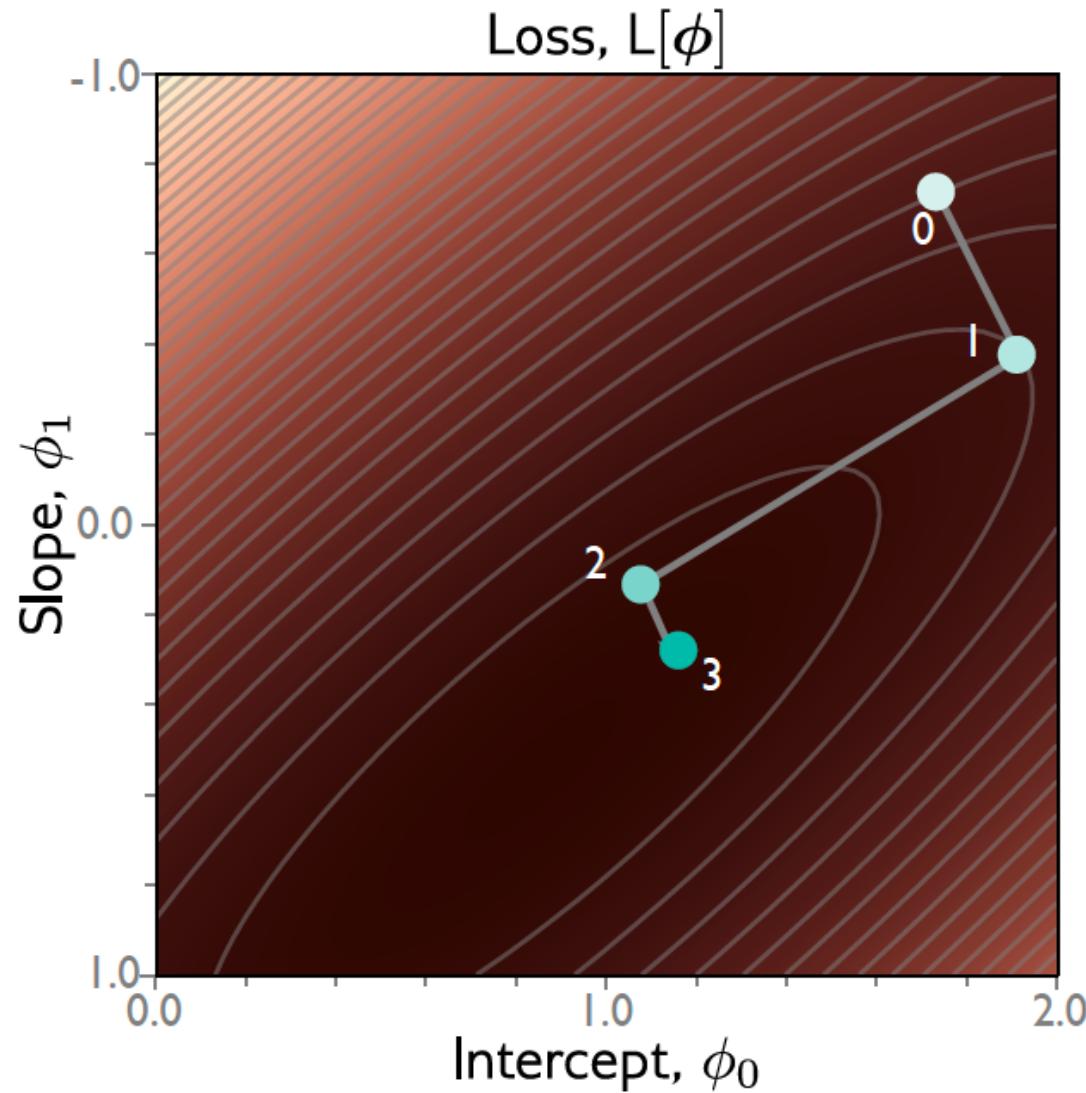
$$\phi \leftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

$\alpha$  = step size

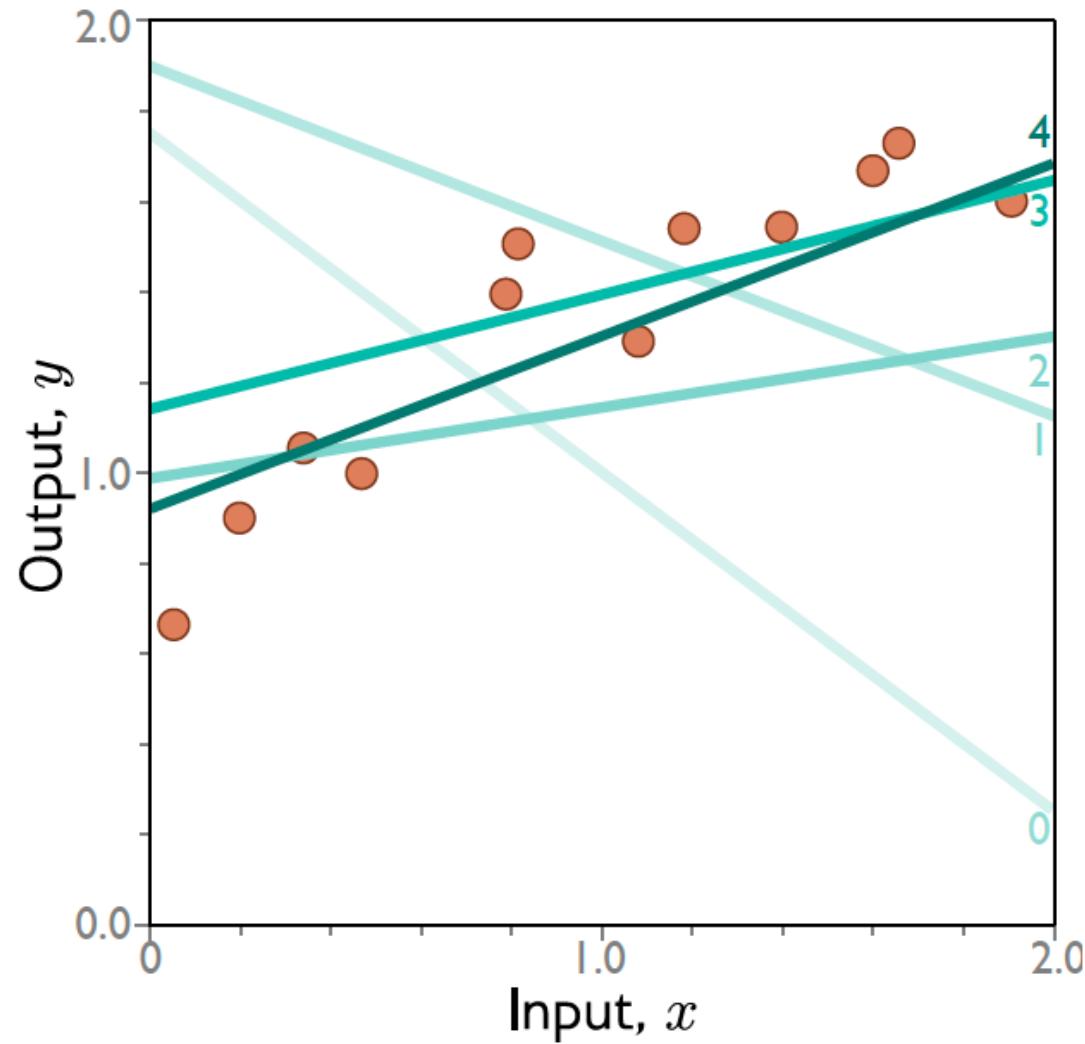
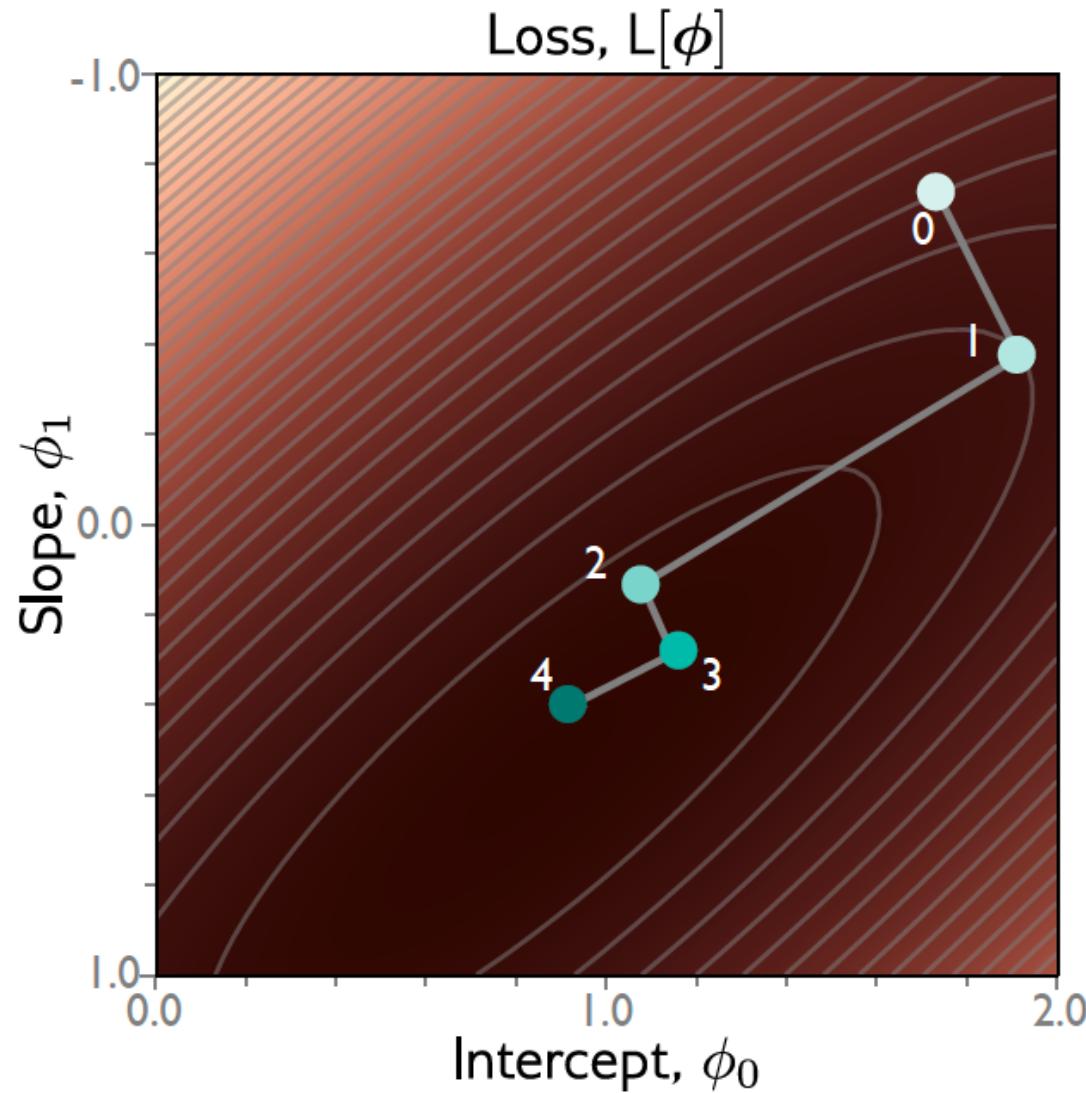
# Gradient descent



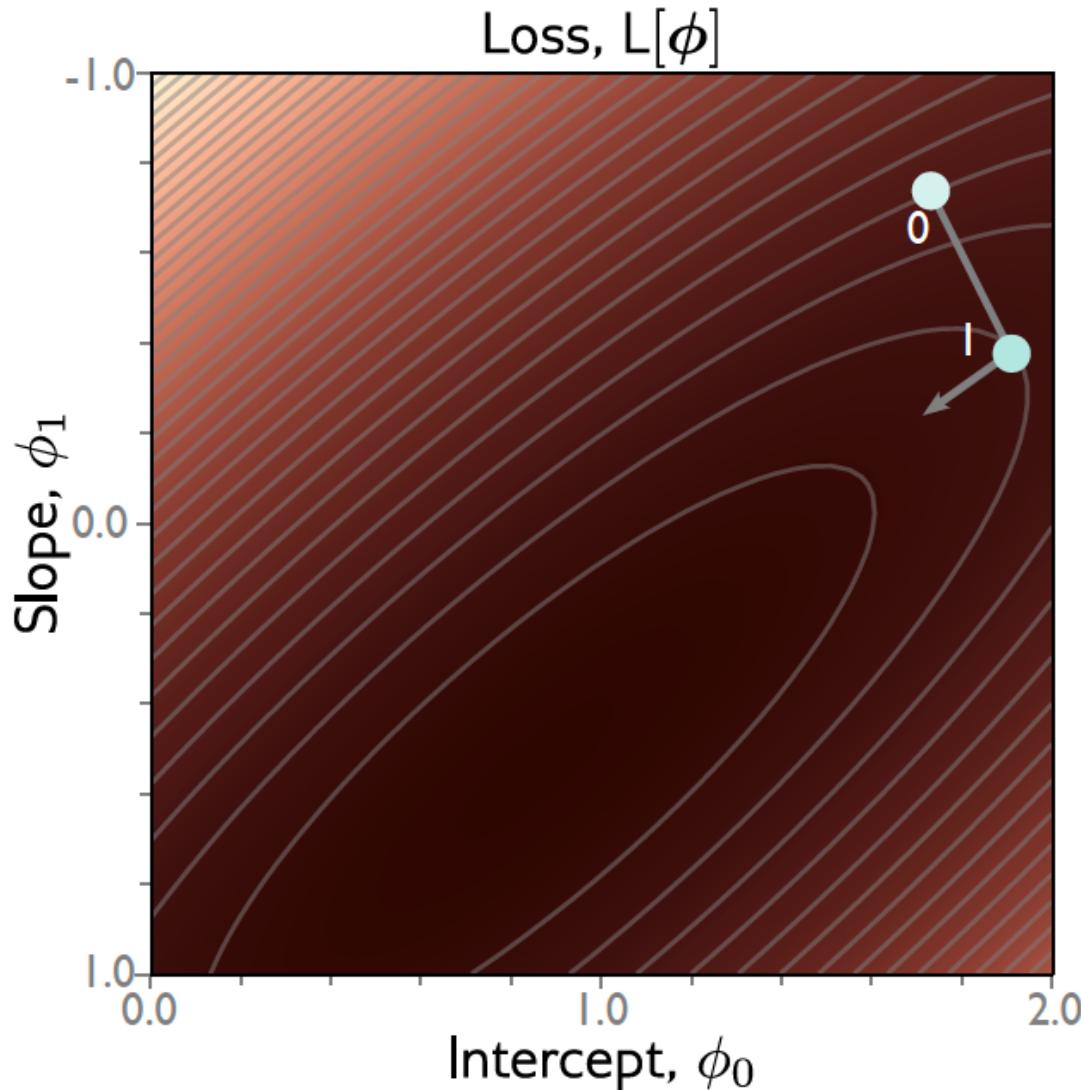
# Gradient descent



# Gradient descent



# Line Search



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

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

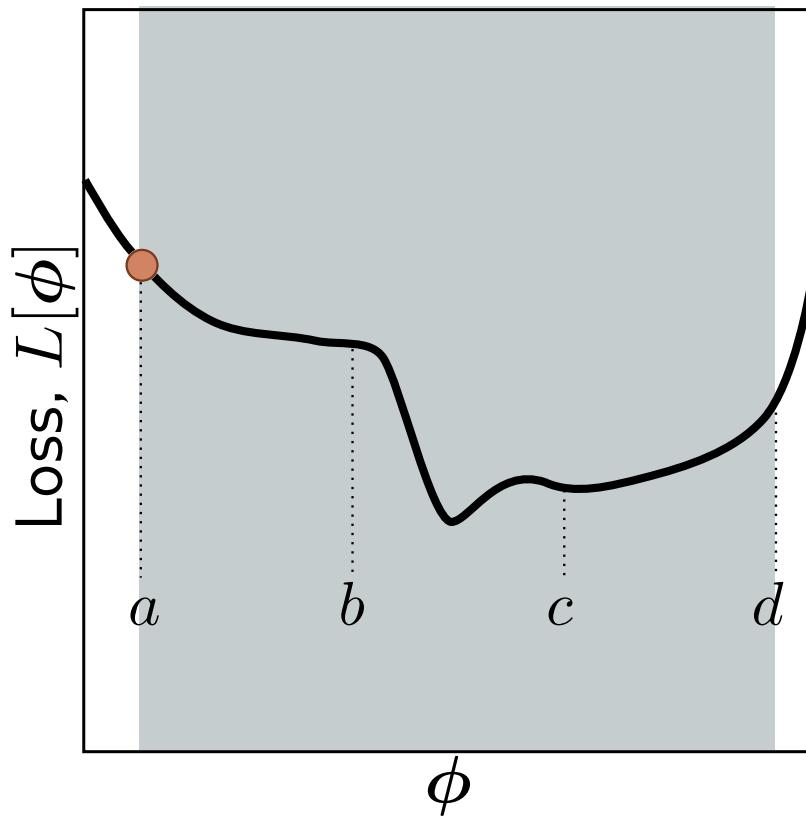
$$\frac{\partial \ell_i}{\partial \phi} = \begin{bmatrix} \frac{\partial \ell_i}{\partial \phi_0} \\ \frac{\partial \ell_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$$

Step 2: Update parameters according to rule

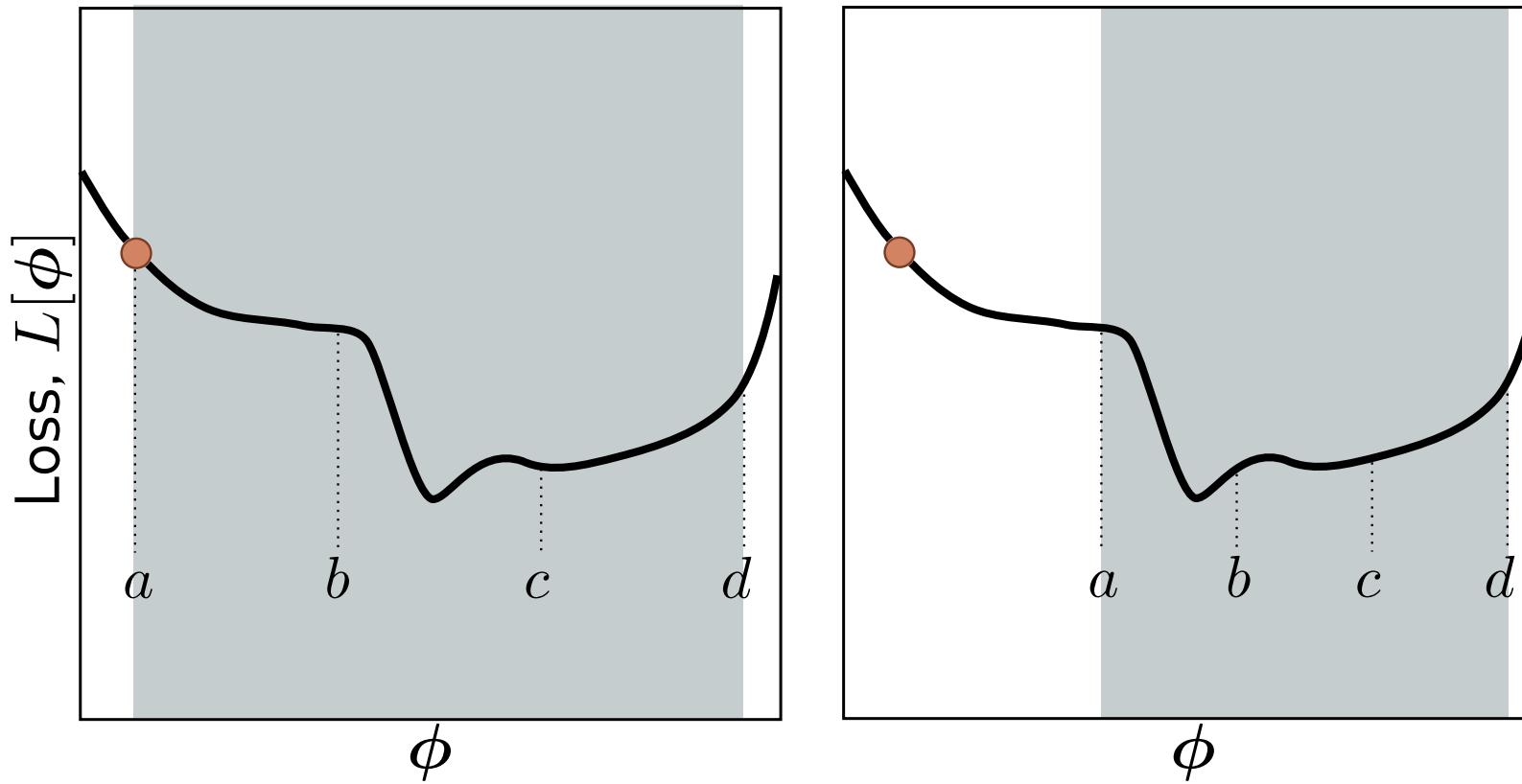
$$\phi \leftarrow \phi - \alpha \frac{\partial L}{\partial \phi}$$

$\alpha$  = step size

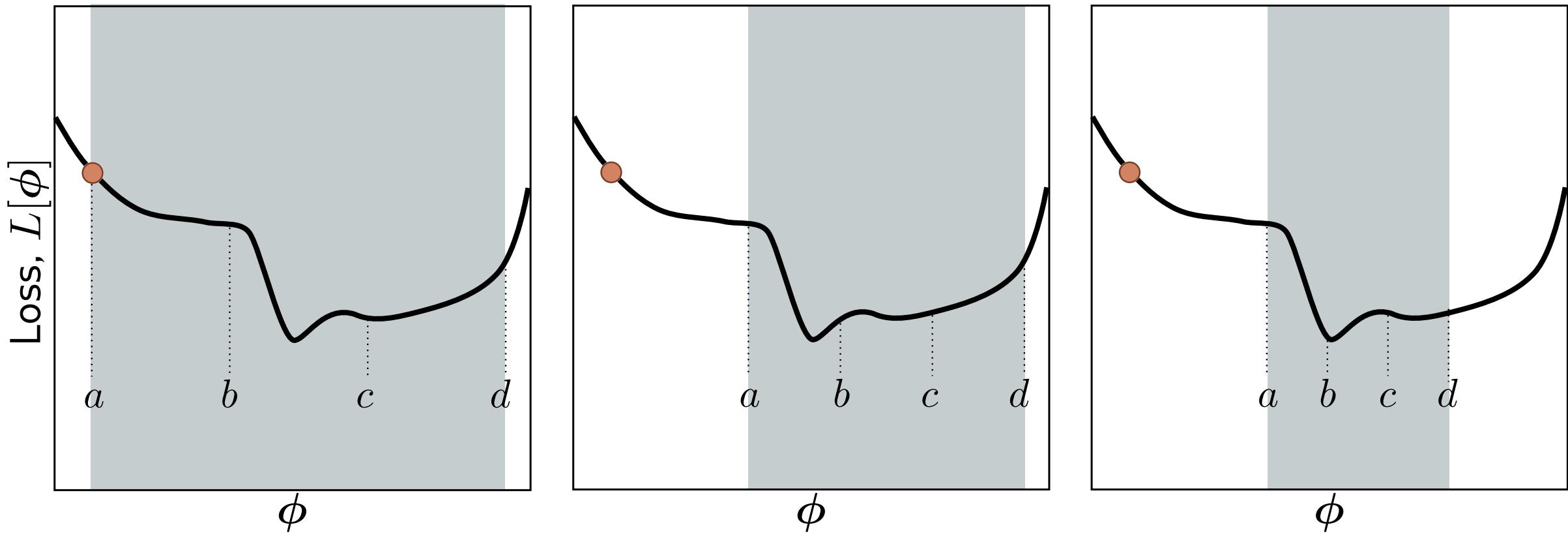
# Line Search (bracketing)



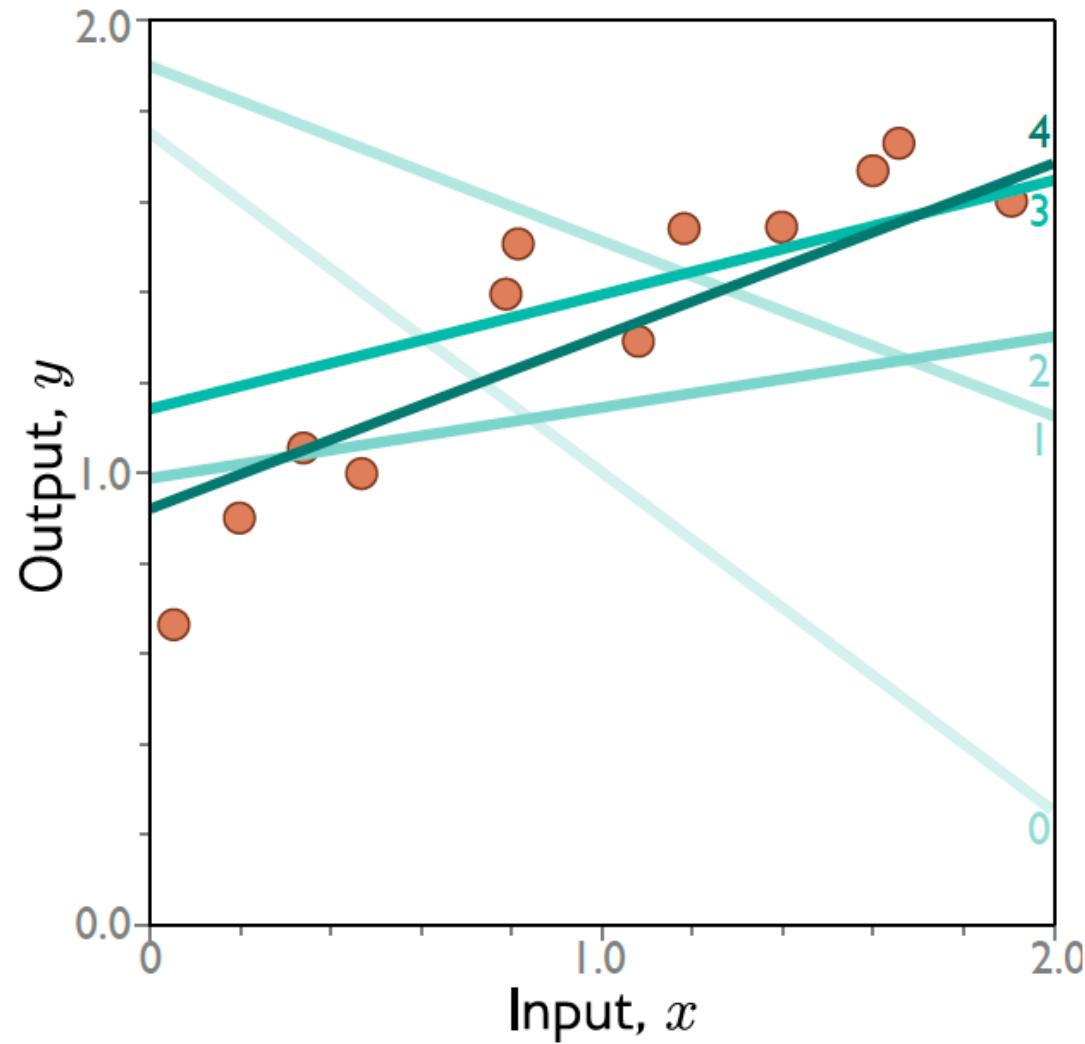
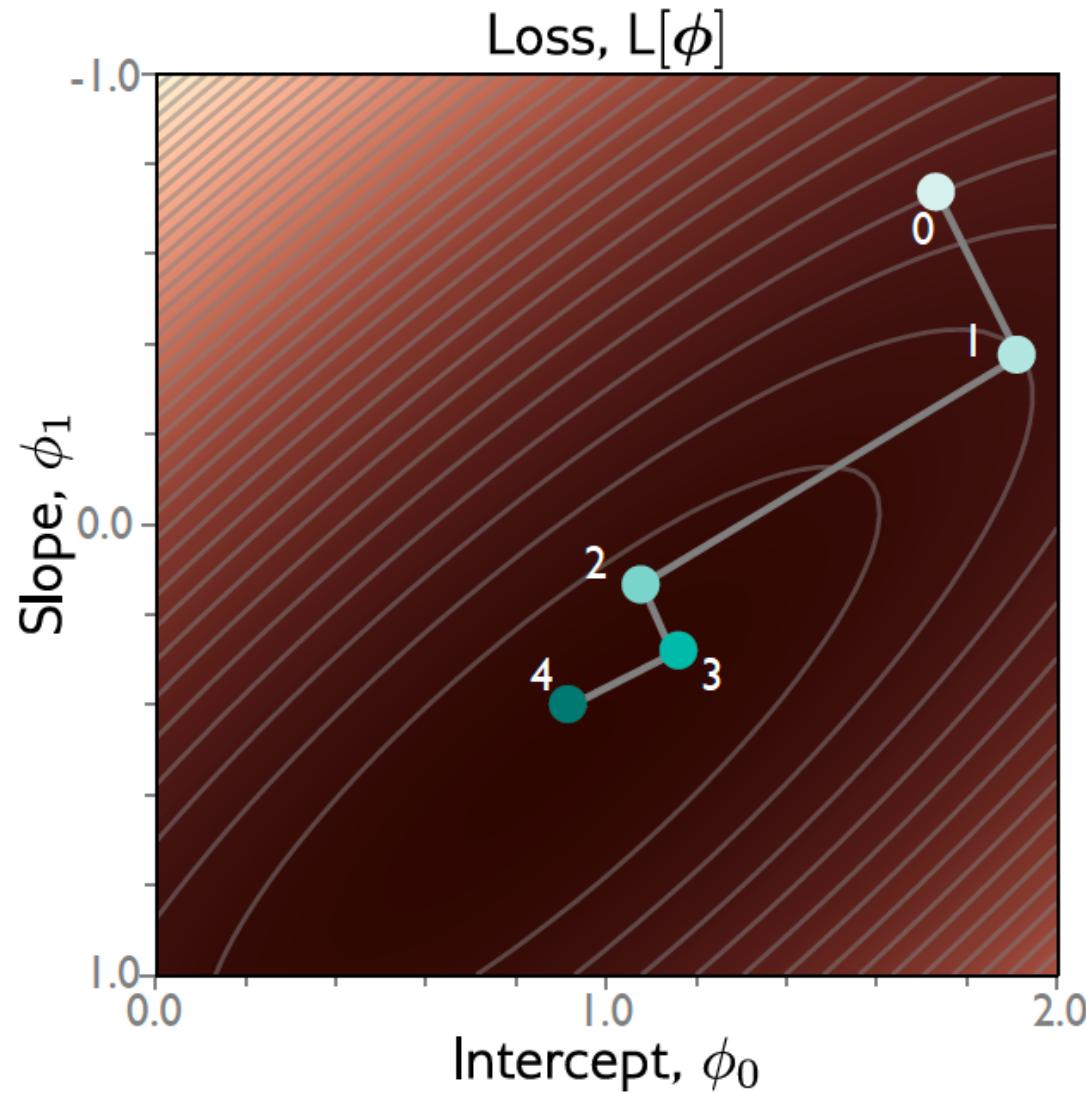
# Line Search (bracketing)



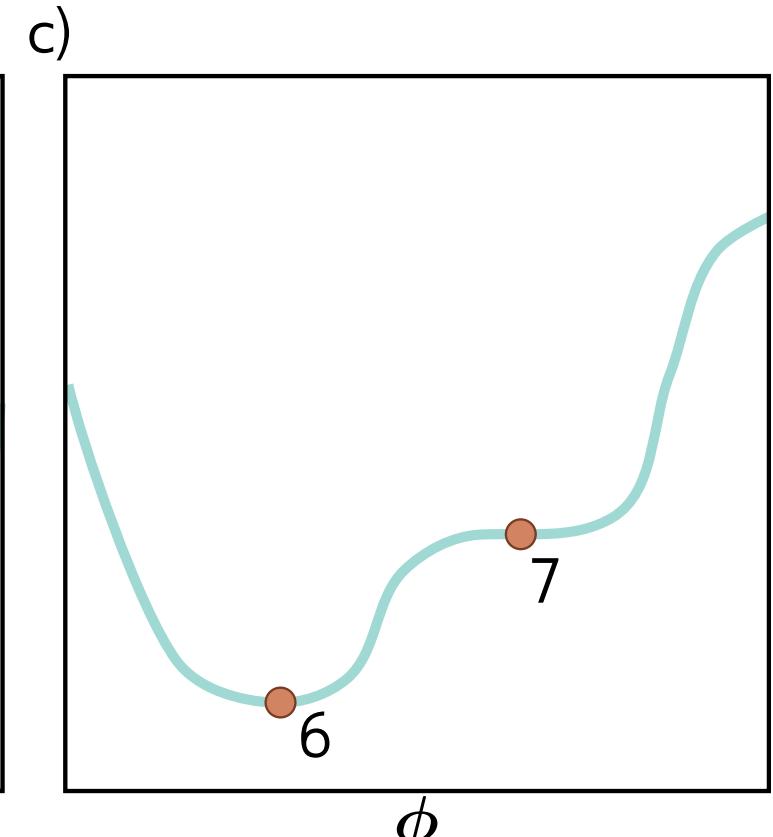
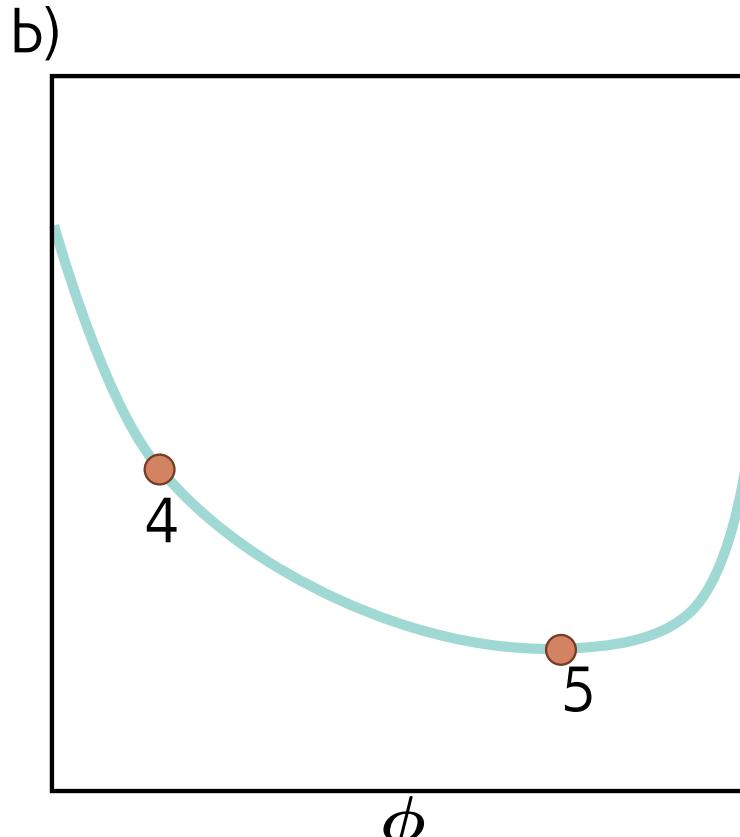
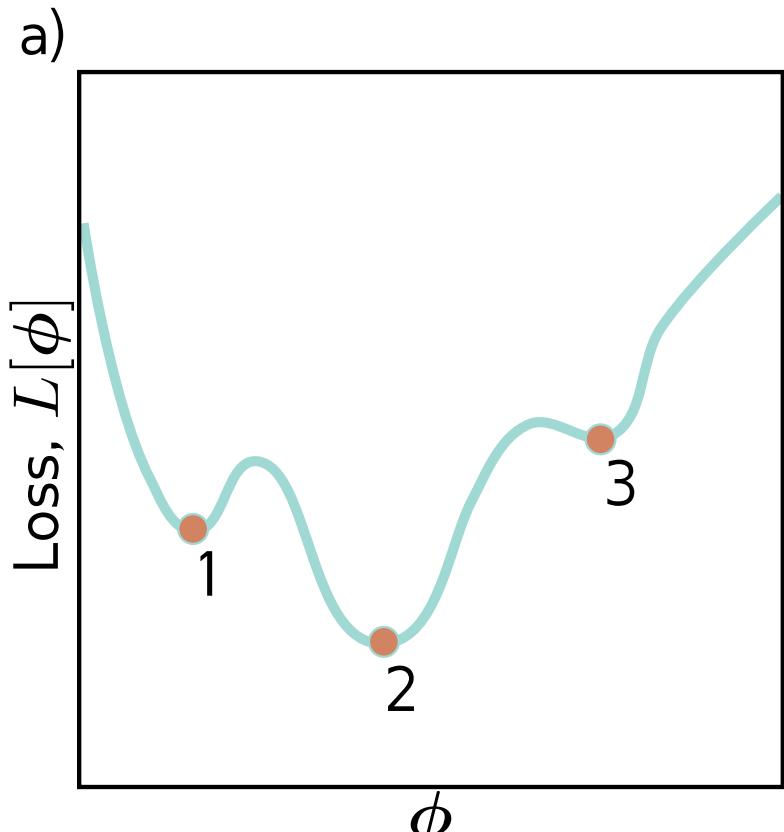
# Line Search (bracketing)



# Gradient descent



# Convex problems

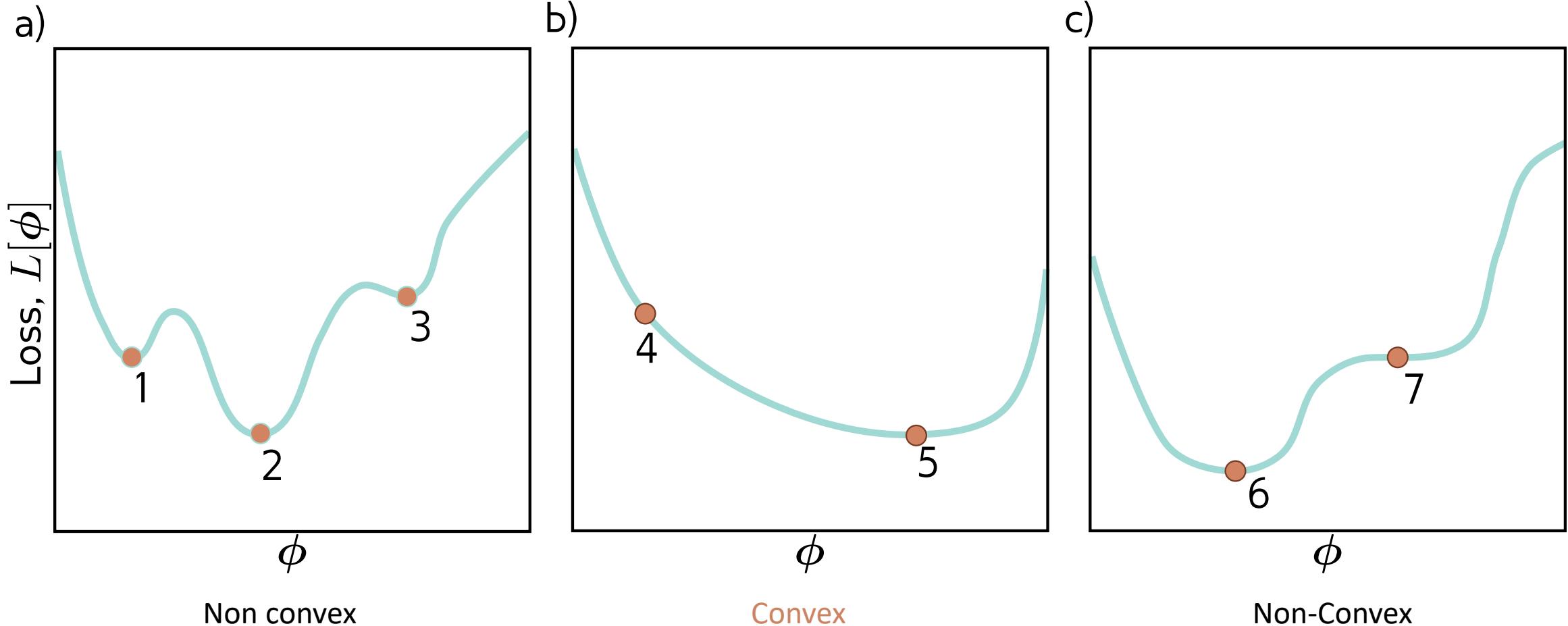


Non convex

Convex

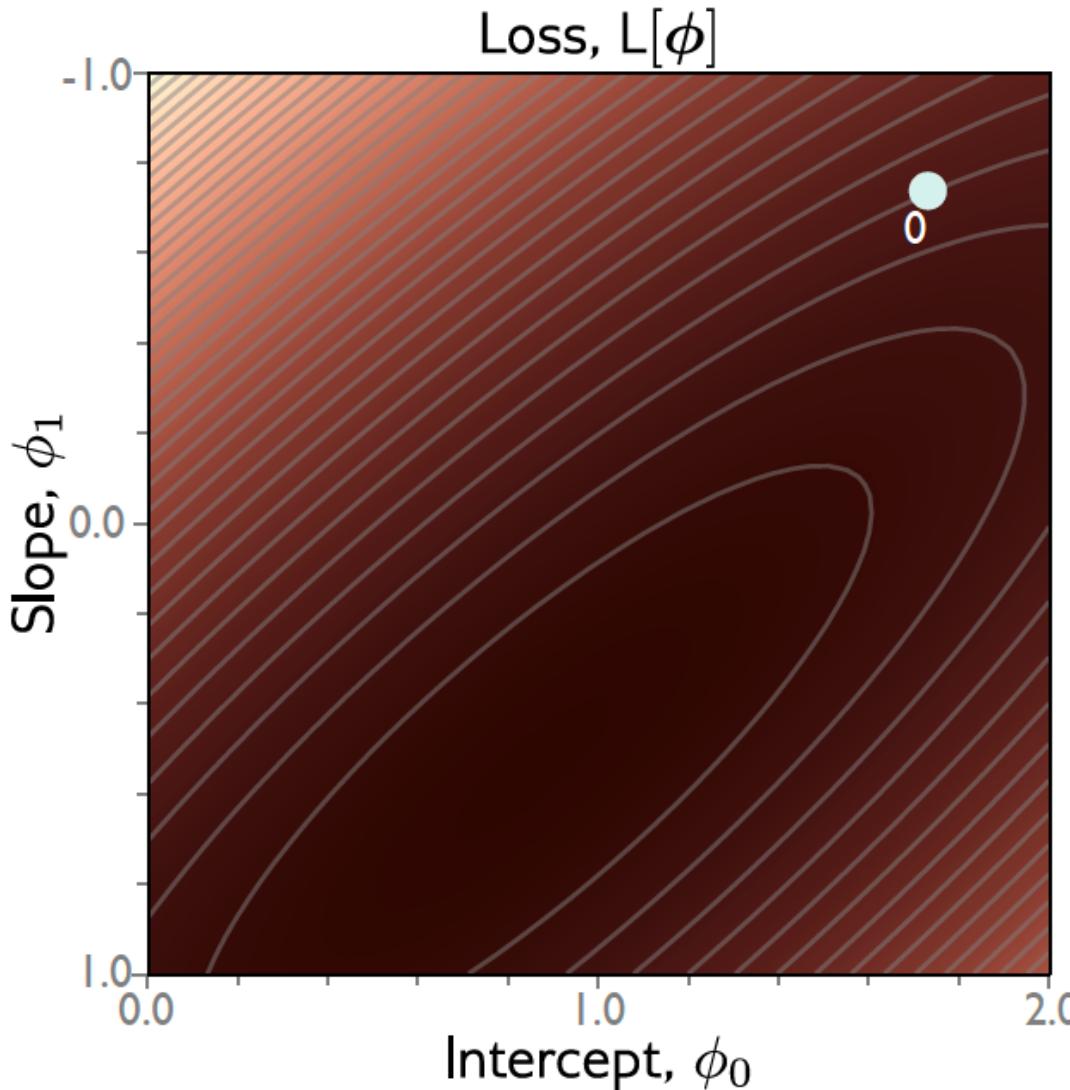
Non-Convex

# Convex problems



Test for convexity is that 2<sup>nd</sup> derivative is positive everywhere

# Convexity in higher dimensions



Test for convexity is that determinant of Hessian (2<sup>nd</sup> derivative matrix) is positive everywhere.

$$\mathbf{H}[\phi] = \begin{bmatrix} \frac{\partial^2 L}{\partial \phi_0^2} & \frac{\partial^2 L}{\partial \phi_0 \partial \phi_1} \\ \frac{\partial^2 L}{\partial \phi_1 \partial \phi_0} & \frac{\partial^2 L}{\partial \phi_1^2} \end{bmatrix}$$

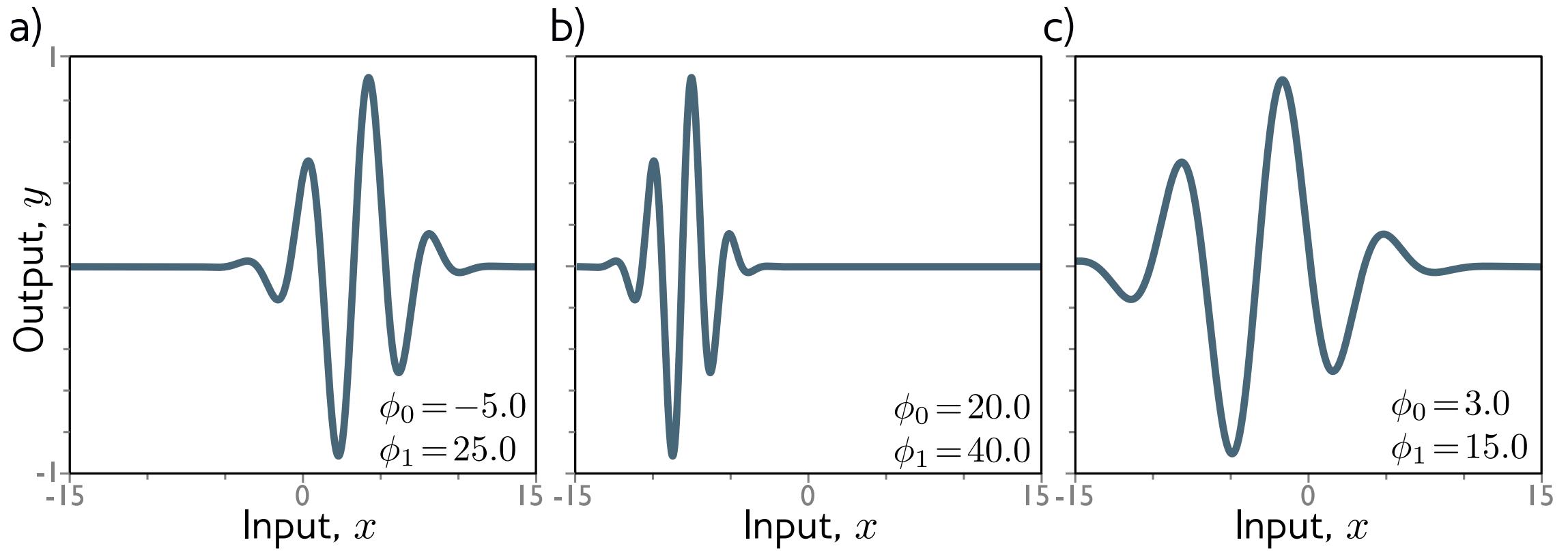
$$\mathbf{H}[\phi] = \frac{\partial^2 L}{\partial \phi_0^2} \frac{\partial^2 L}{\partial \phi_1^2} - \frac{\partial^2 L}{\partial \phi_0 \partial \phi_1} \frac{\partial^2 L}{\partial \phi_1 \partial \phi_0}$$

# Fitting models

- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

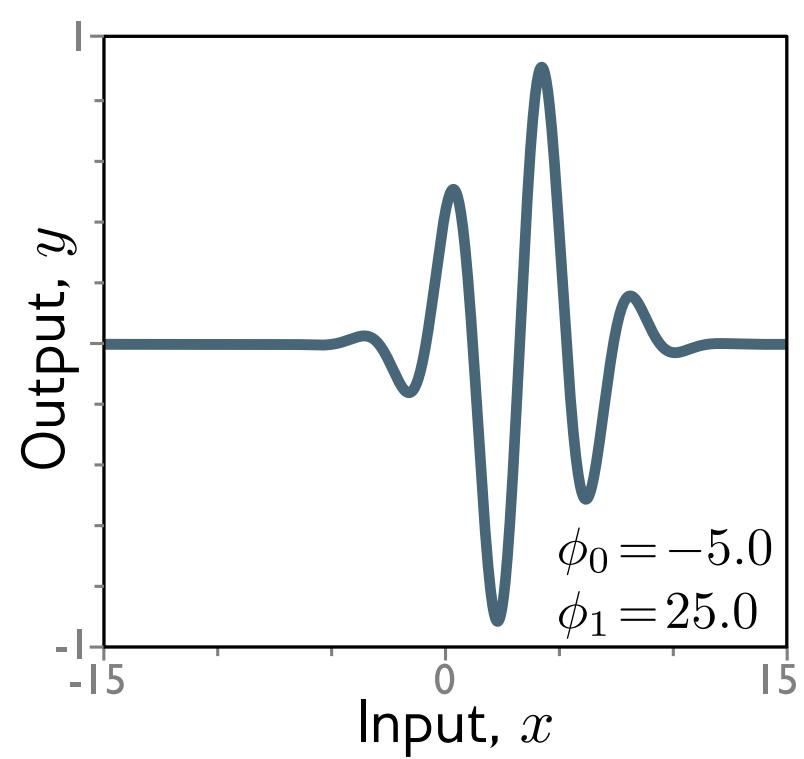
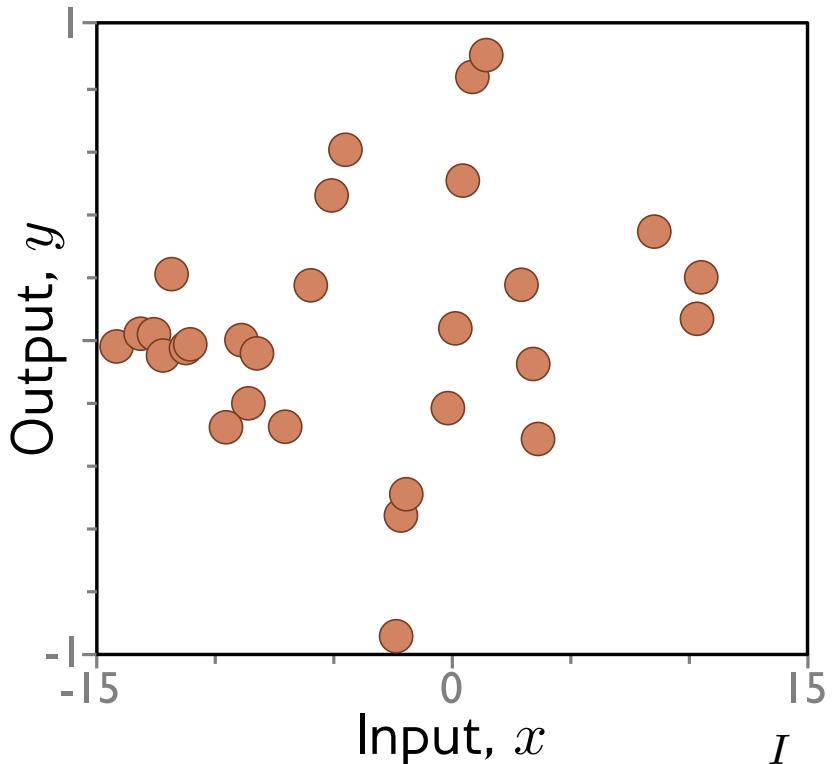
# Gabor model

$$f[x, \phi] = \sin[\phi_0 + 0.06 \cdot \phi_1 x] \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{8.0}\right)$$

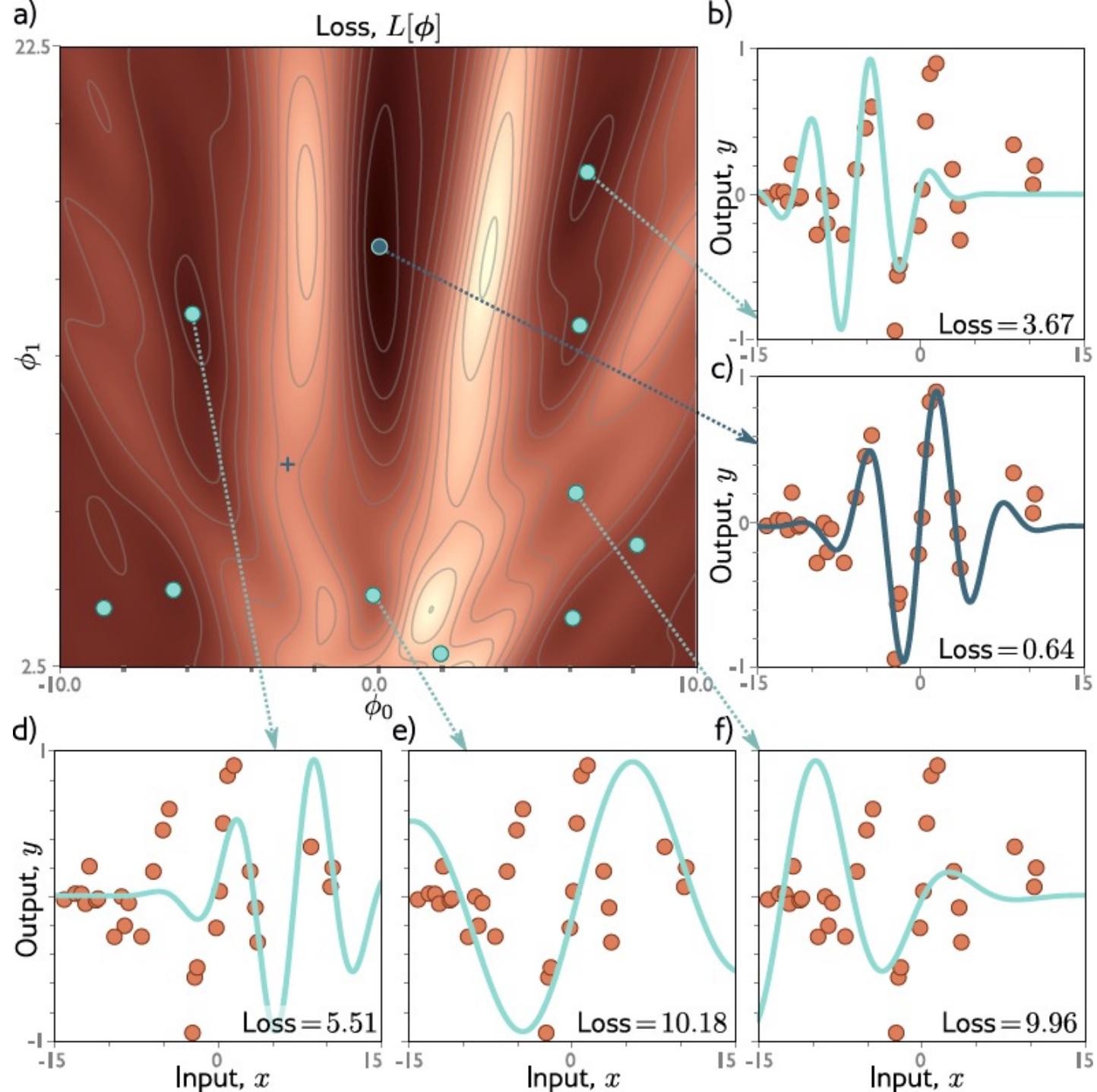


# Gabor model

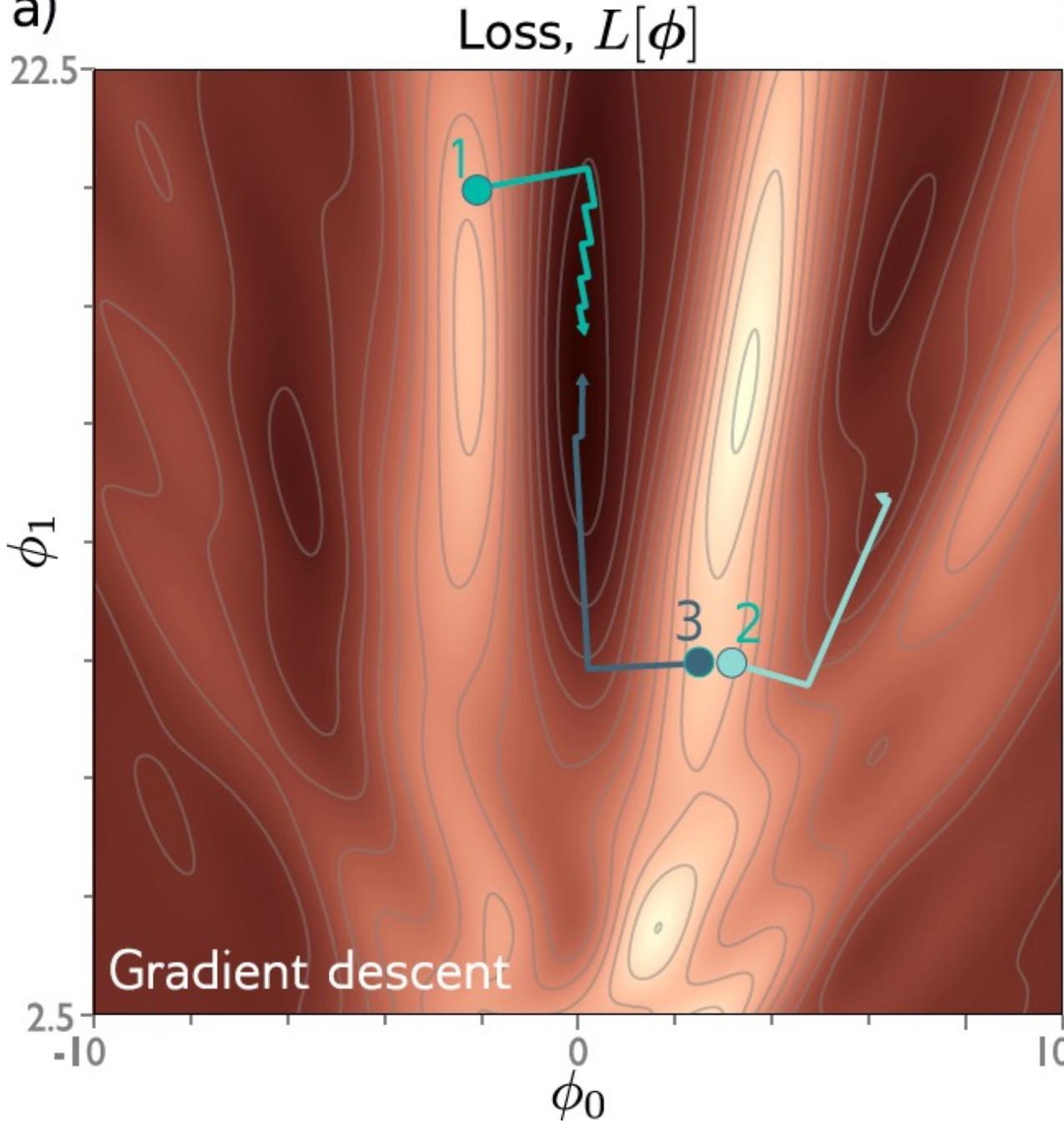
$$f[x, \phi] = \sin[\phi_0 + 0.06 \cdot \phi_1 x] \cdot \exp\left(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{8.0}\right)$$



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



a)

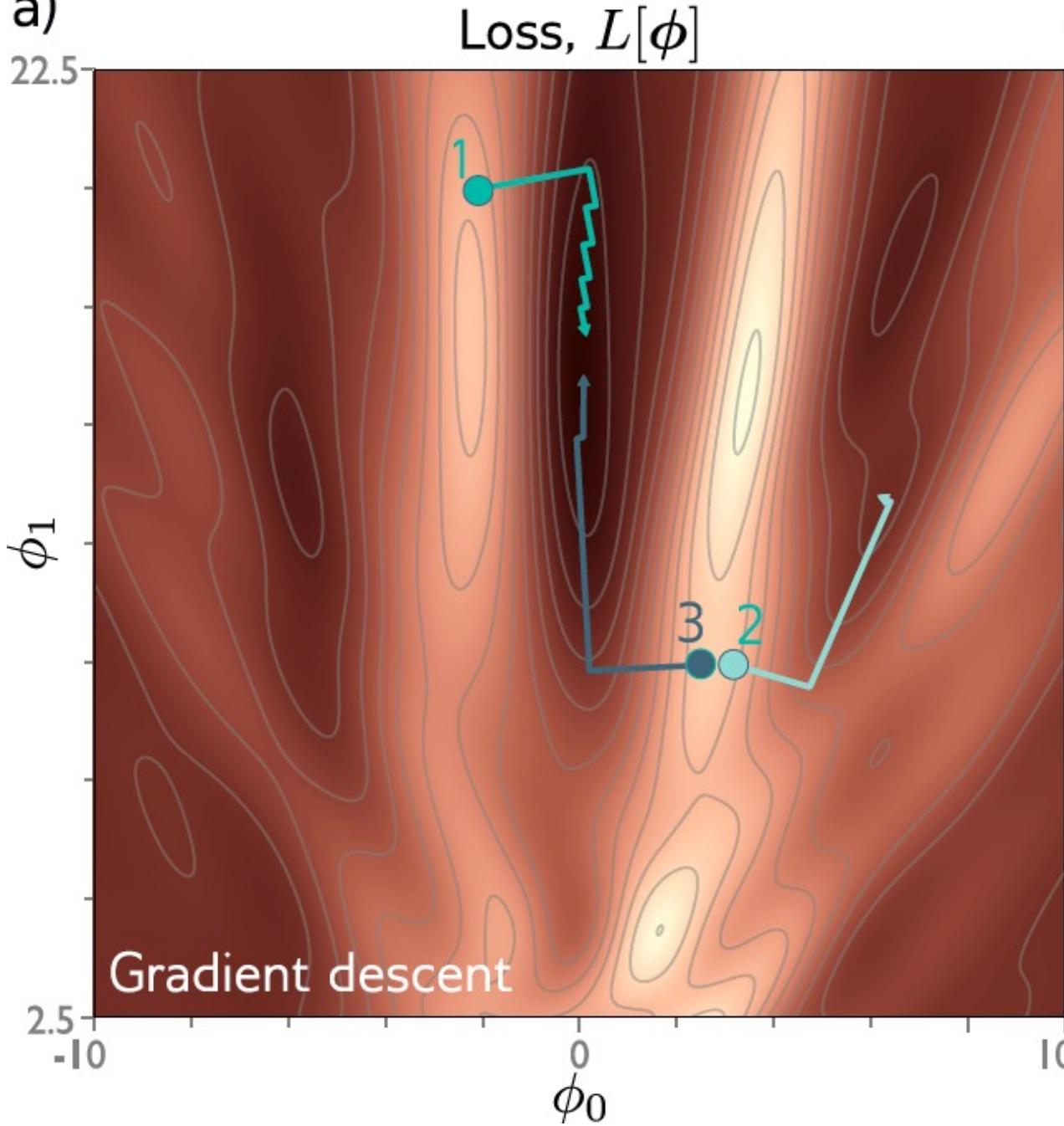


- Gradient descent gets to the global minimum if we start in the right “valley”
- Otherwise, descent to a local minimum
- Or get stuck near a saddle point

# Fitting models

- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

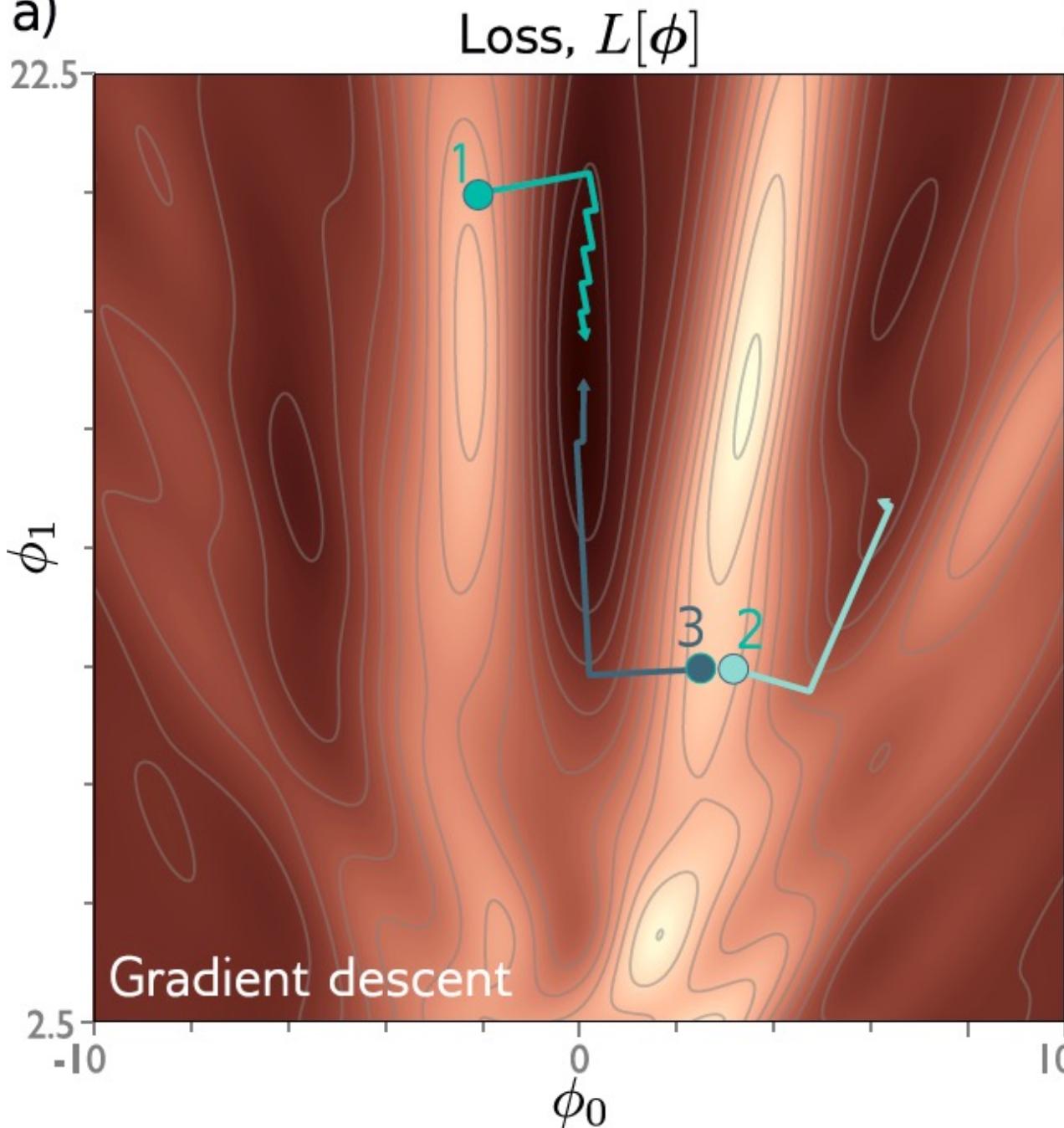
a)



IDEA: add noise

- Stochastic gradient descent
- Compute gradient based on only a subset of points – a **mini-batch**
- Work through dataset sampling without replacement
- One pass though the data is called an **epoch**

a)



Stochastic gradient descent

Before (full batch descent)

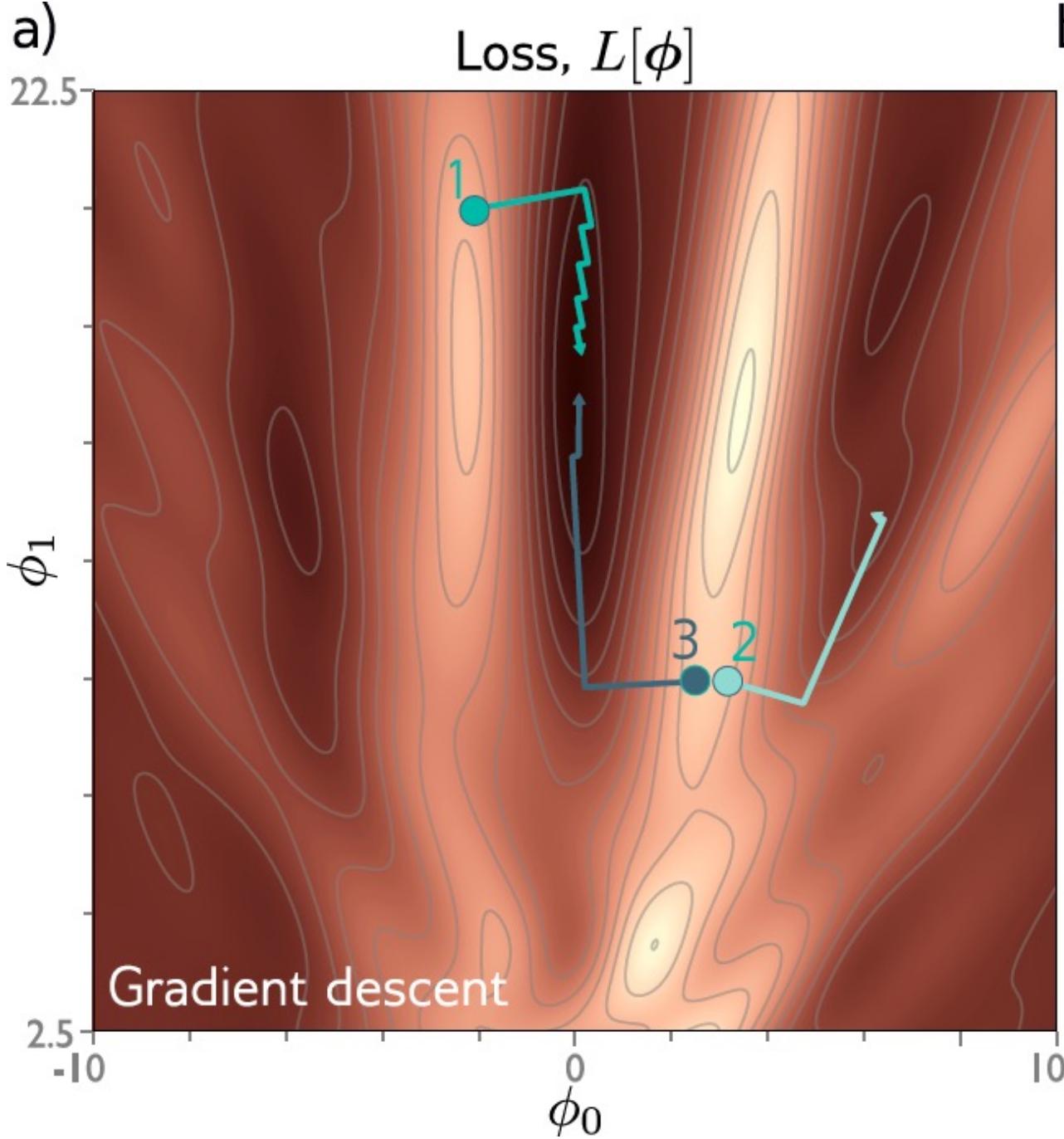
$$\phi_{t+1} \leftarrow \phi_t - \alpha \sum_{i=1}^I \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

After (SGD)

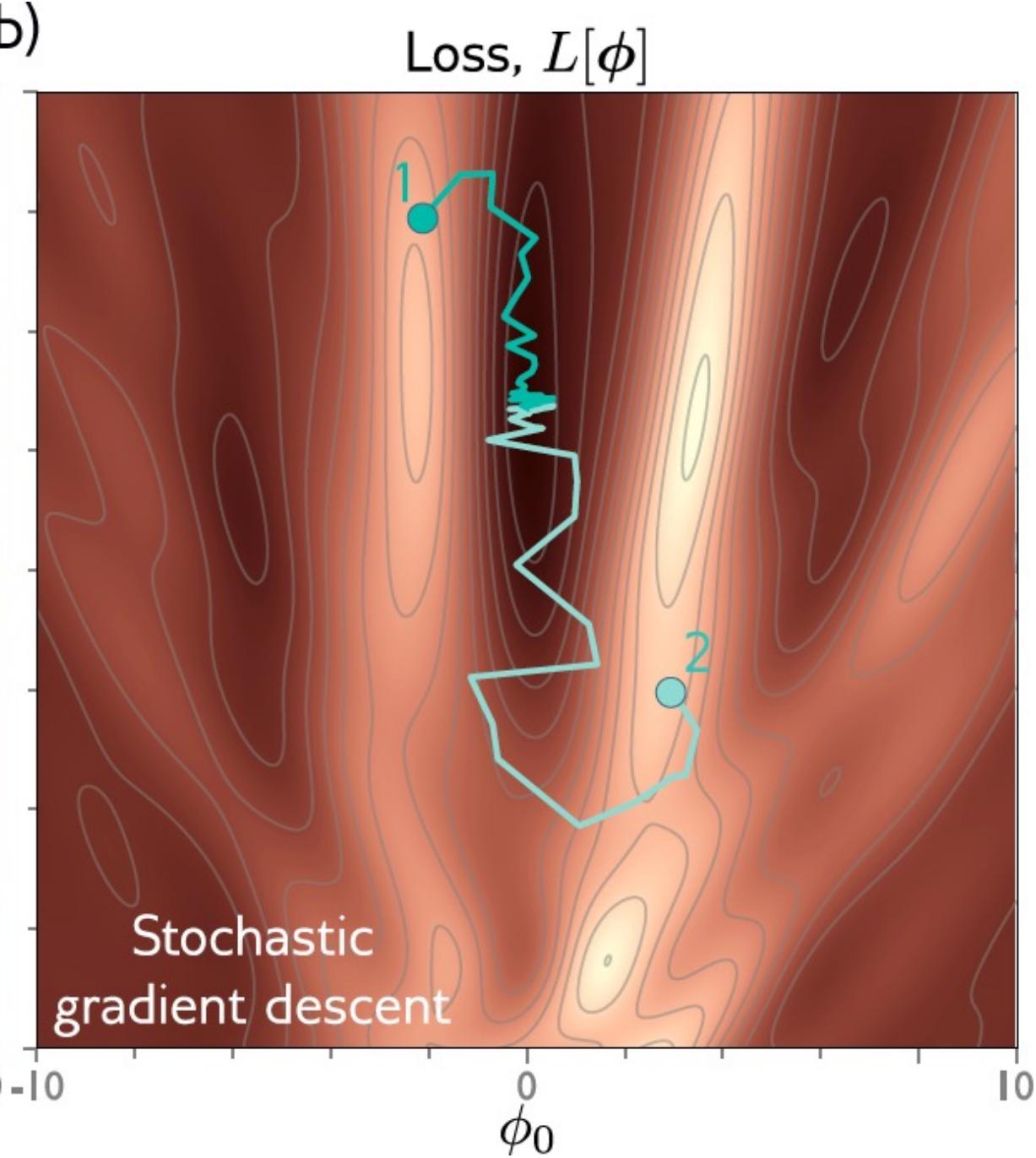
$$\phi_{t+1} \leftarrow \phi_t - \alpha \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi},$$

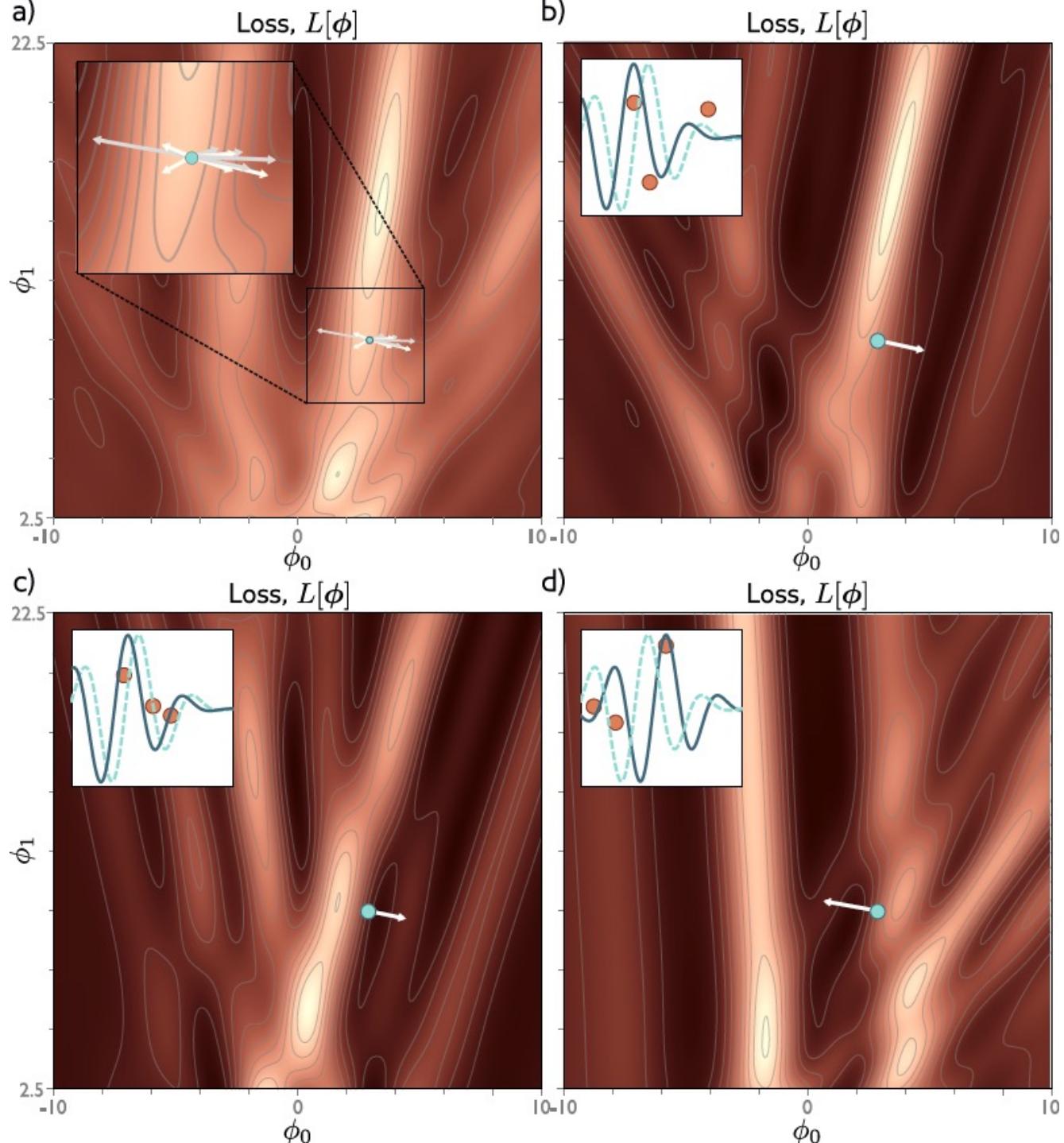
Fixed learning rate  $\alpha$

a)



b)





# Properties of SGD

- Can escape from local minima
- Adds noise, but still sensible updates as based on part of data
- Uses all data equally
- Less computationally expensive
- Seems to find better solutions
- Doesn't converge in traditional sense
- Learning rate schedule – decrease learning rate over time

# Fitting models

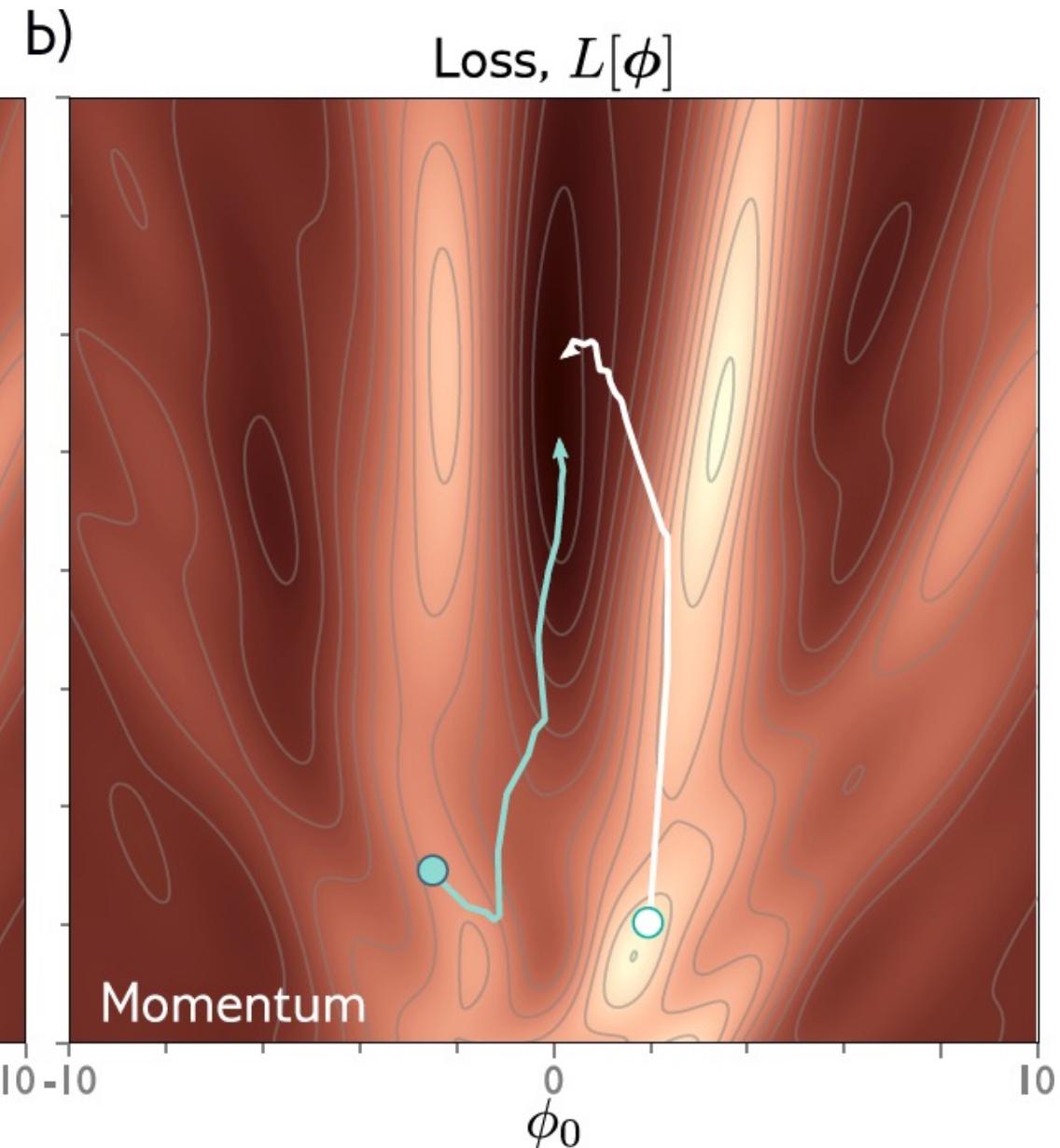
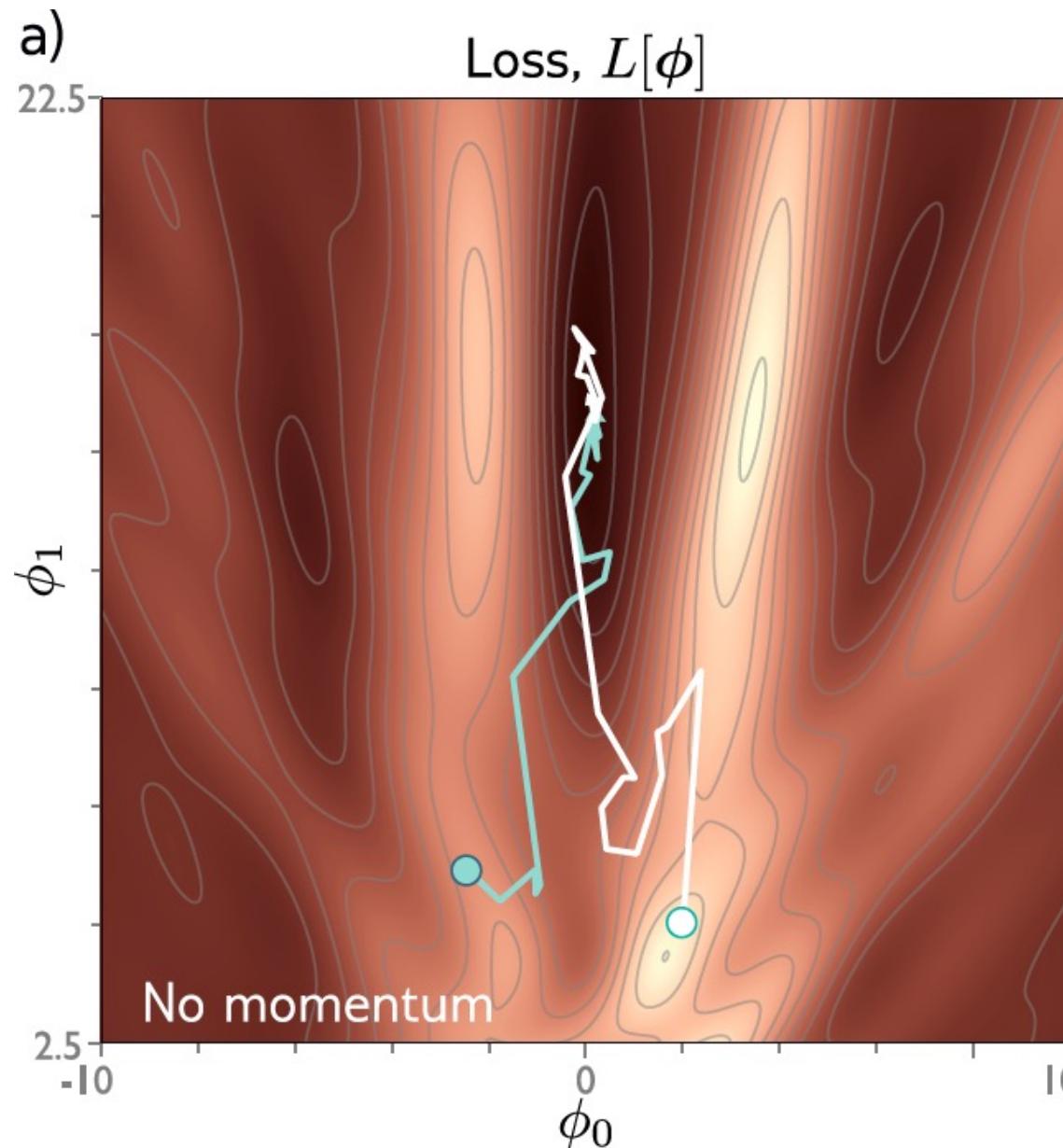
- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

# Momentum

- Weighted sum of this gradient and previous gradient

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi}$$

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \mathbf{m}_{t+1}$$



# Nesterov accelerated momentum

- Momentum is kind of like a prediction of where we are going

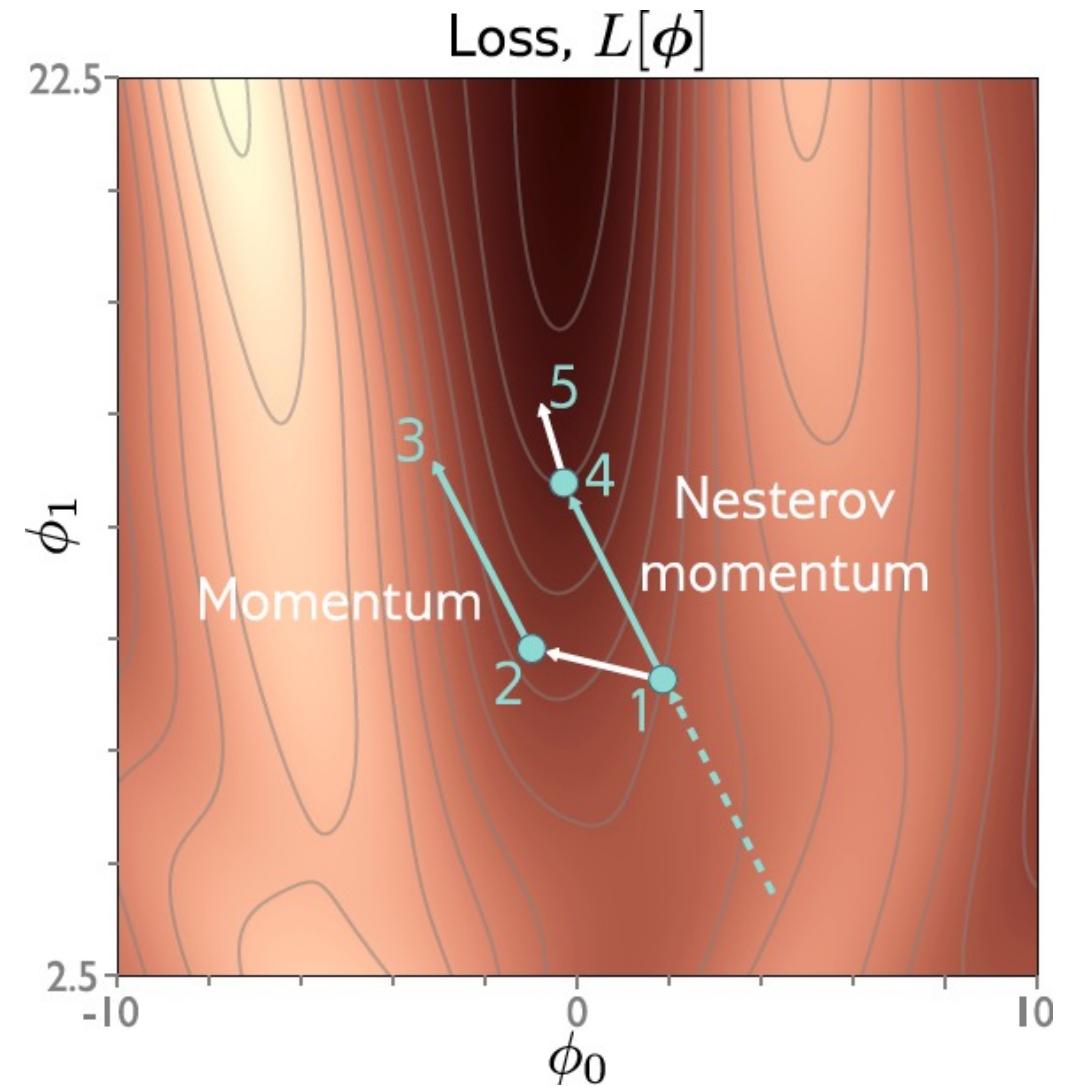
$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t]}{\partial \phi}$$

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \mathbf{m}_{t+1}$$

- Move in the predicted direction, THEN, measure the gradient

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \sum_{i \in \mathcal{B}_t} \frac{\partial \ell_i[\phi_t - \alpha \cdot \mathbf{m}_t]}{\partial \phi}$$

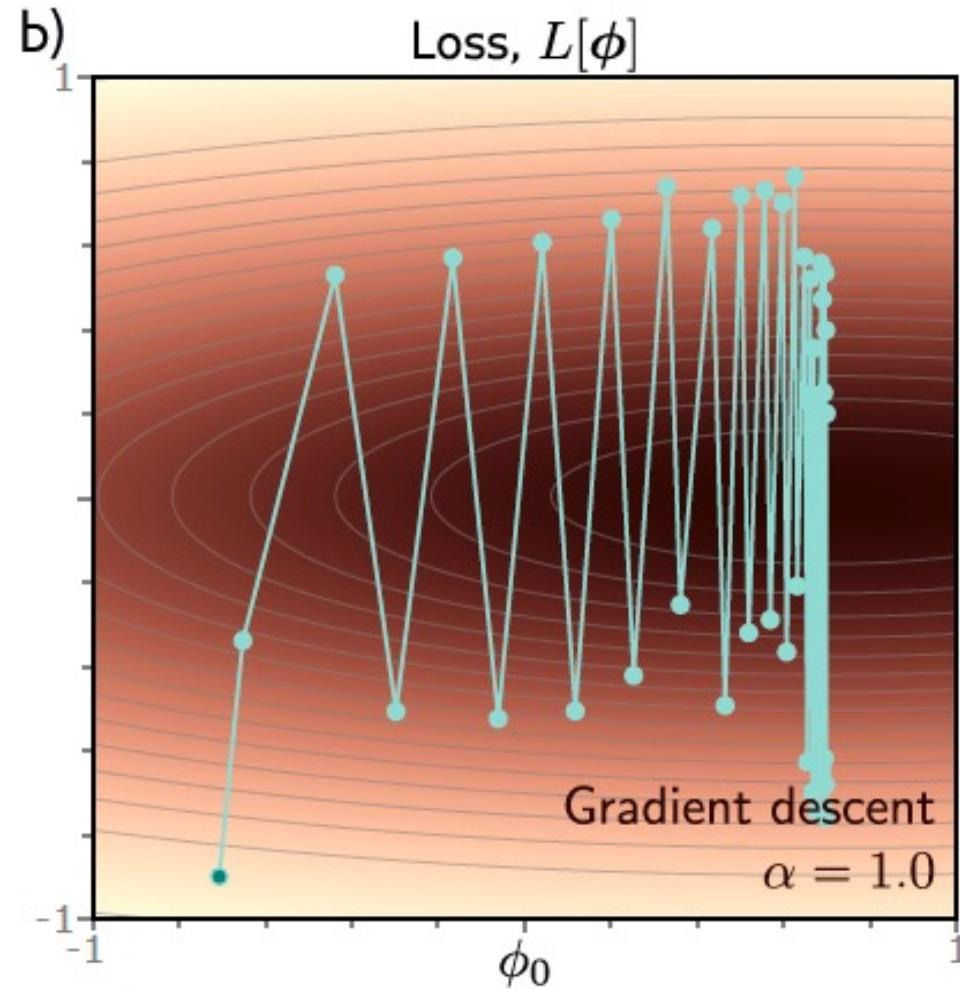
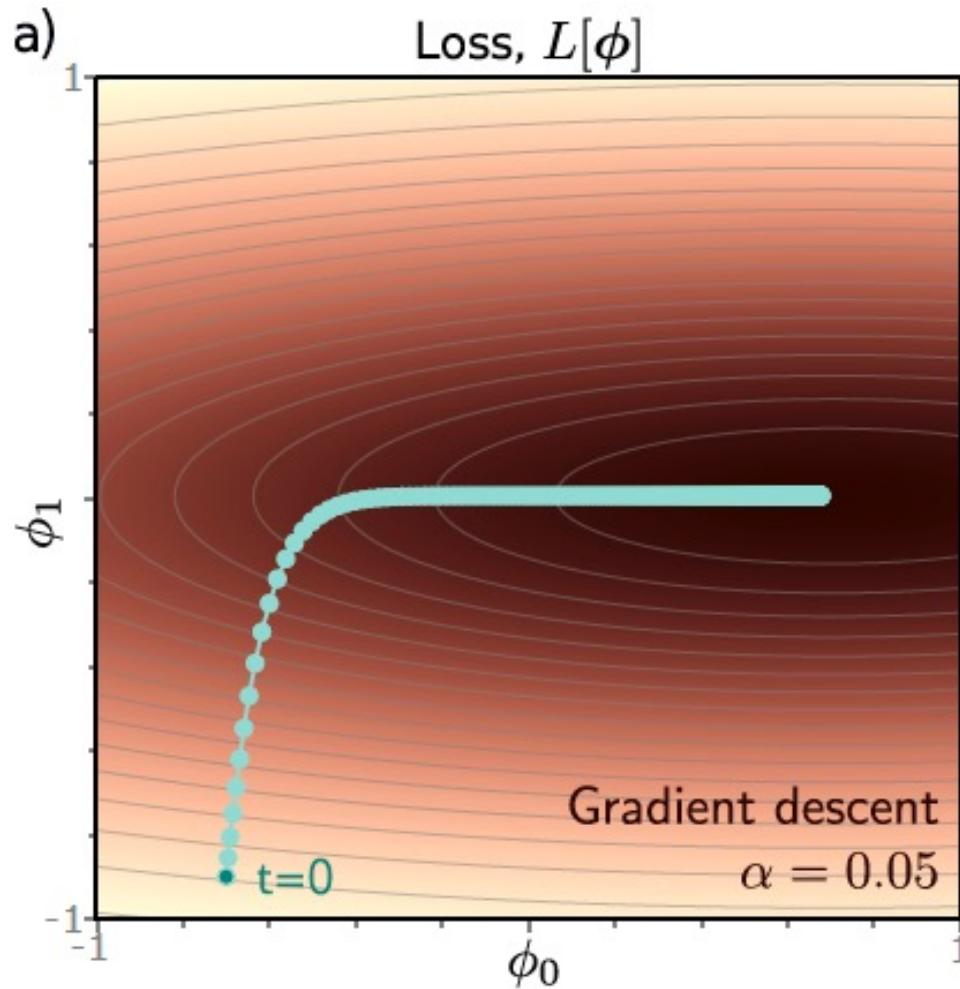
$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \mathbf{m}_{t+1}$$



# Fitting models

- Maths overview
- Gradient descent algorithm
- Linear regression example
- Gabor model example
- Stochastic gradient descent
- Momentum
- Adam

# Adaptive moment estimation (Adam)



# Normalized gradients

- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\phi_t]}{\partial \phi}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\phi_t]^2}{\partial \phi}$$

- Normalize:

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

# Normalized gradients

- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\phi_t]}{\partial \phi}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\phi_t]^2}{\partial \phi}$$

- Normalize:

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon}$$

$$\mathbf{m}_{t+1} = \begin{bmatrix} 3.0 \\ -2.0 \\ 5.0 \end{bmatrix}$$

$$\mathbf{v}_{t+1} = \begin{bmatrix} 9.0 \\ 4.0 \\ 25.0 \end{bmatrix}$$

$$\frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}} + \epsilon} = \begin{bmatrix} 1.0 \\ -1.0 \\ 1.0 \end{bmatrix}$$

# Normalized gradients

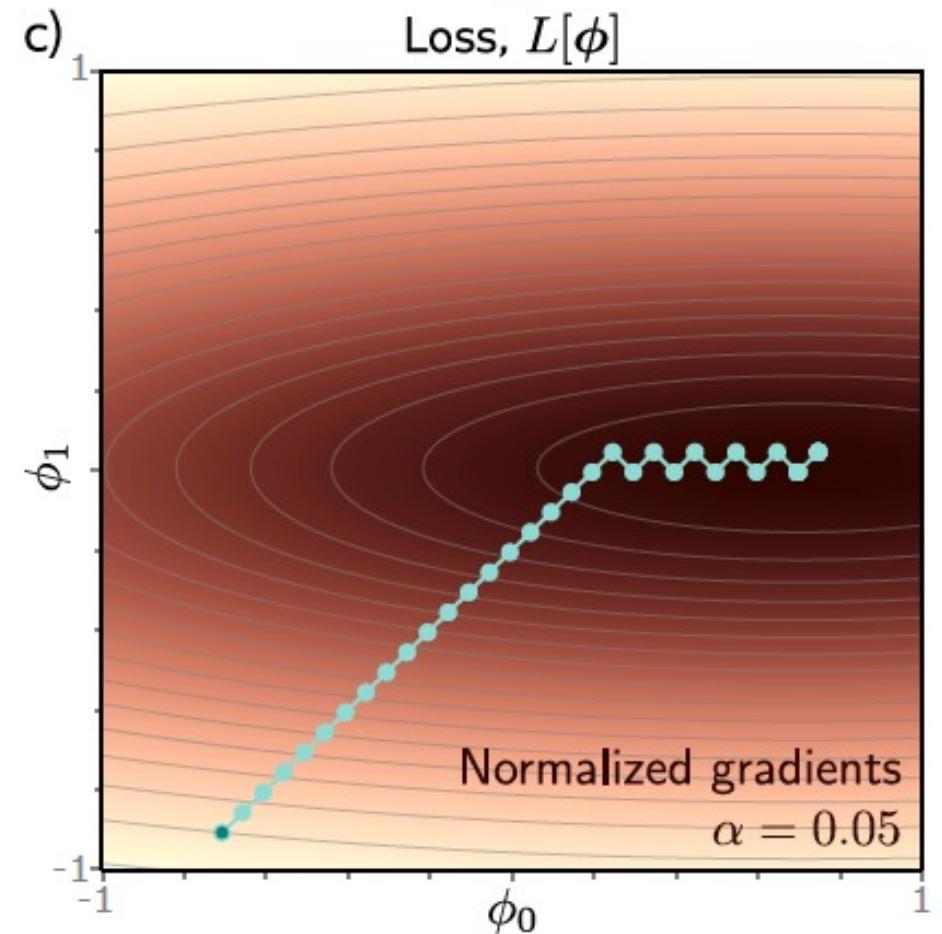
- Measure mean and pointwise squared gradient

$$\mathbf{m}_{t+1} \leftarrow \frac{\partial L[\phi_t]}{\partial \phi}$$

$$\mathbf{v}_{t+1} \leftarrow \frac{\partial L[\phi_t]^2}{\partial \phi}$$

- Normalize:

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \frac{\mathbf{m}_{t+1}}{\sqrt{\mathbf{v}_{t+1}}} + \epsilon$$



# Adaptive moment estimation (Adam)

- Compute mean and pointwise squared gradients with momentum

$$\mathbf{m}_{t+1} \leftarrow \beta \cdot \mathbf{m}_t + (1 - \beta) \frac{\partial L[\phi_t]}{\partial \phi}$$

$$\mathbf{v}_{t+1} \leftarrow \gamma \cdot \mathbf{v}_t + (1 - \gamma) \left( \frac{\partial L[\phi_t]}{\partial \phi} \right)^2$$

- Moderate near start of the sequence

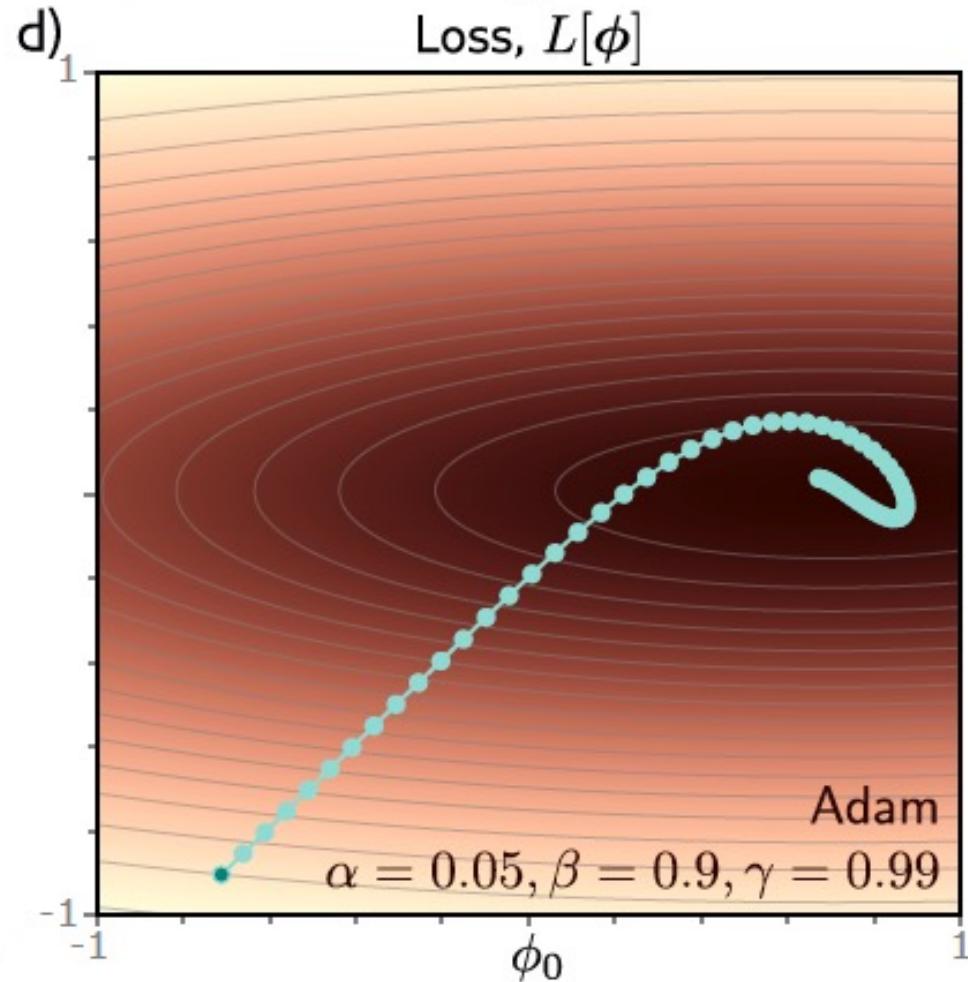
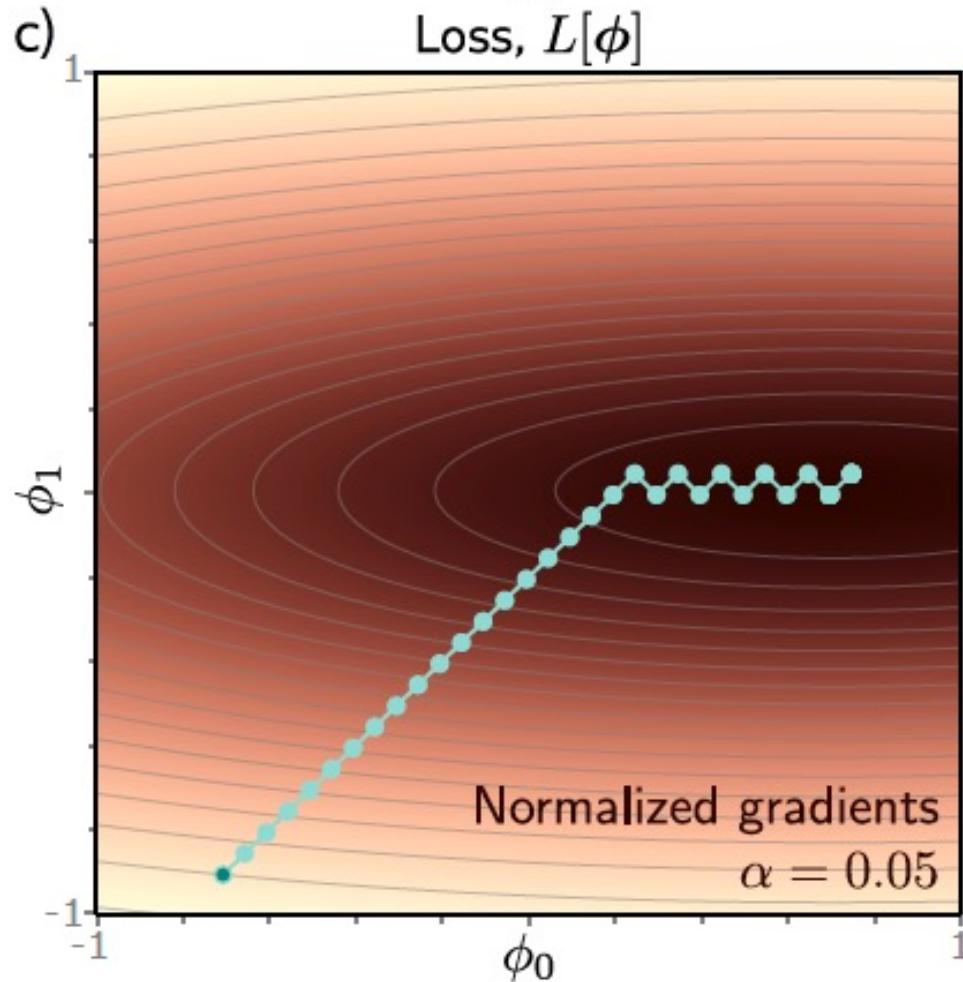
$$\tilde{\mathbf{m}}_{t+1} \leftarrow \frac{\mathbf{m}_{t+1}}{1 - \beta^{t+1}}$$

$$\tilde{\mathbf{v}}_{t+1} \leftarrow \frac{\mathbf{v}_{t+1}}{1 - \gamma^{t+1}}$$

- Update the parameters

$$\phi_{t+1} \leftarrow \phi_t - \alpha \cdot \frac{\tilde{\mathbf{m}}_{t+1}}{\sqrt{\tilde{\mathbf{v}}_{t+1}} + \epsilon}$$

# Adaptive moment estimation (Adam)



# Hyperparameters

- Choice of learning algorithm
- Learning rate
- Momentum



Feedback