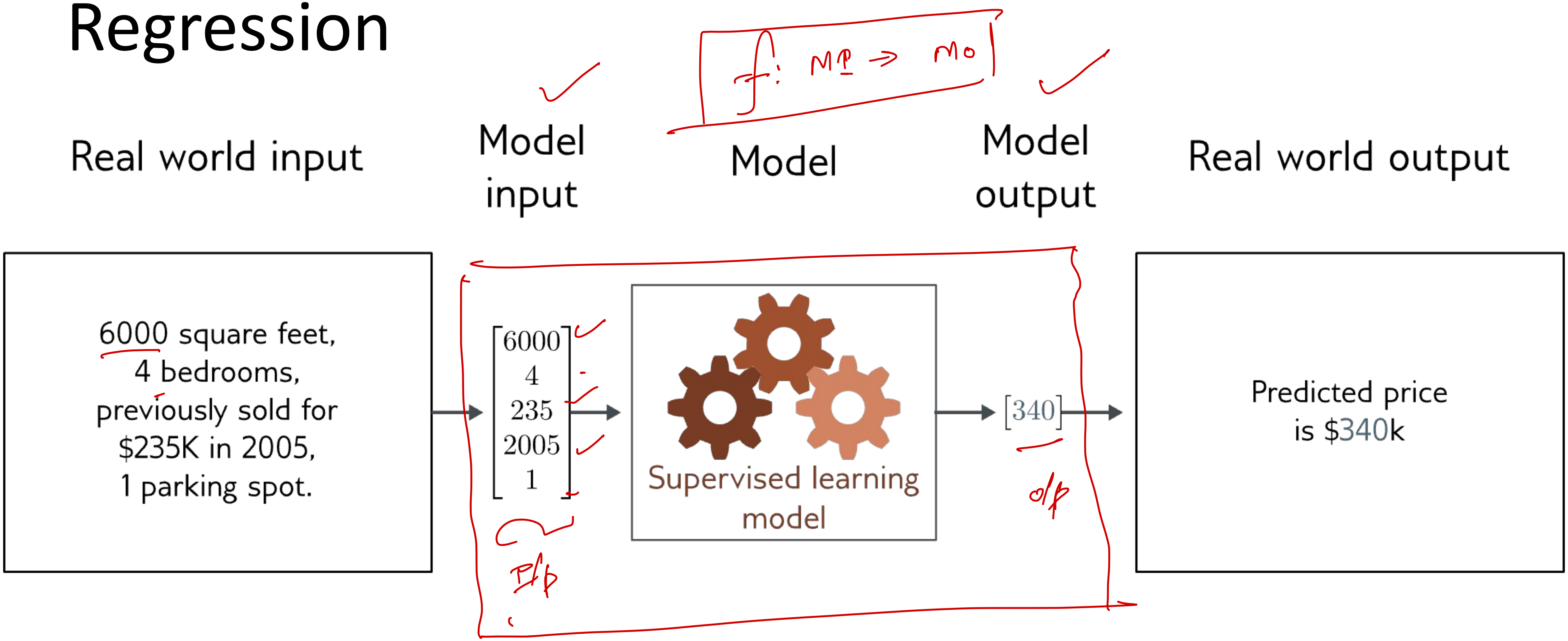
Supervised Learning

January 3rd, 2025

Deep Learning (CS60010)

**Slides adapted from http://udlbook.com*

Regression



- Univariate regression problem (one output, real value)

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a mathematical equation

- Example:
  - Input is age and mileage of second hand Toyota Prius
  - Output is estimated price of car

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a mathematical equation

Training

vs

- Computing the outputs from the inputs = inference
- Model also includes parameters

[Your model is completely known]

- Parameters affect outcome of equation

parameters

in terms of parameters

$$y = w_0 + w_1 x_1$$

Supervised learning overview

Model
↓
parameters
(use training)

- Supervised learning model = mapping from one or more inputs to one or more outputs
- ~~Model is a mathematical equation~~
- Model is a family of equations → one param eqn [Training]
- Computing the outputs from the inputs = inference
- Model also includes parameters
- Parameters affect outcome of equation
- Training a model = finding parameters that predict outputs "well" from inputs for a training dataset of input/output pairs

$\boxed{y = w_0 + w_1 x}$

$y = 1 + 2x$

$0.1 + 0.3x$

$0.7 - 1.4x$

Notation:

- Input:

$$\mathbf{x}$$

- Output:

$$\mathbf{y}$$

- Model:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]$$

Variables always Roman letters

Normal = scalar
Bold = vector
Capital Bold = matrix

Functions always square brackets

Normal = returns scalar
Bold = returns vector
Capital Bold = returns matrix

Notation example:

- Input:

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{mileage} \end{bmatrix}$$

Structured or tabular data

- Output:

$$y = \begin{bmatrix} \text{price} \end{bmatrix}$$

scalar

- Model:

$$y = \mathrm{f}[\mathbf{x}]$$

Model

- Parameters: ϕ

- Model : $\mathbf{y} = \mathbf{f}[\mathbf{x}, \phi]$

output   input   parameters

Parameters always Greek letters

$f = w_0 + w_1 x$

$\phi = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$

i/p

$\boxed{y = w_0 + w_1 x}$

params

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:

$$L\Big[\underbrace{\phi, \mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\Big]$$

we define a loss fn

I i/p o/p pairs

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:

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

Testing

- To test the model, run on a separate test dataset of input / output pairs

- See how well it generalizes to new data

X

don't show the o/p

Example: 1D Linear regression model

- Model:

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{matrix} \longleftarrow \text{y-offset} \\ \longleftarrow \text{slope} \end{matrix}$$

Example: 1D Linear regression model

- Model:

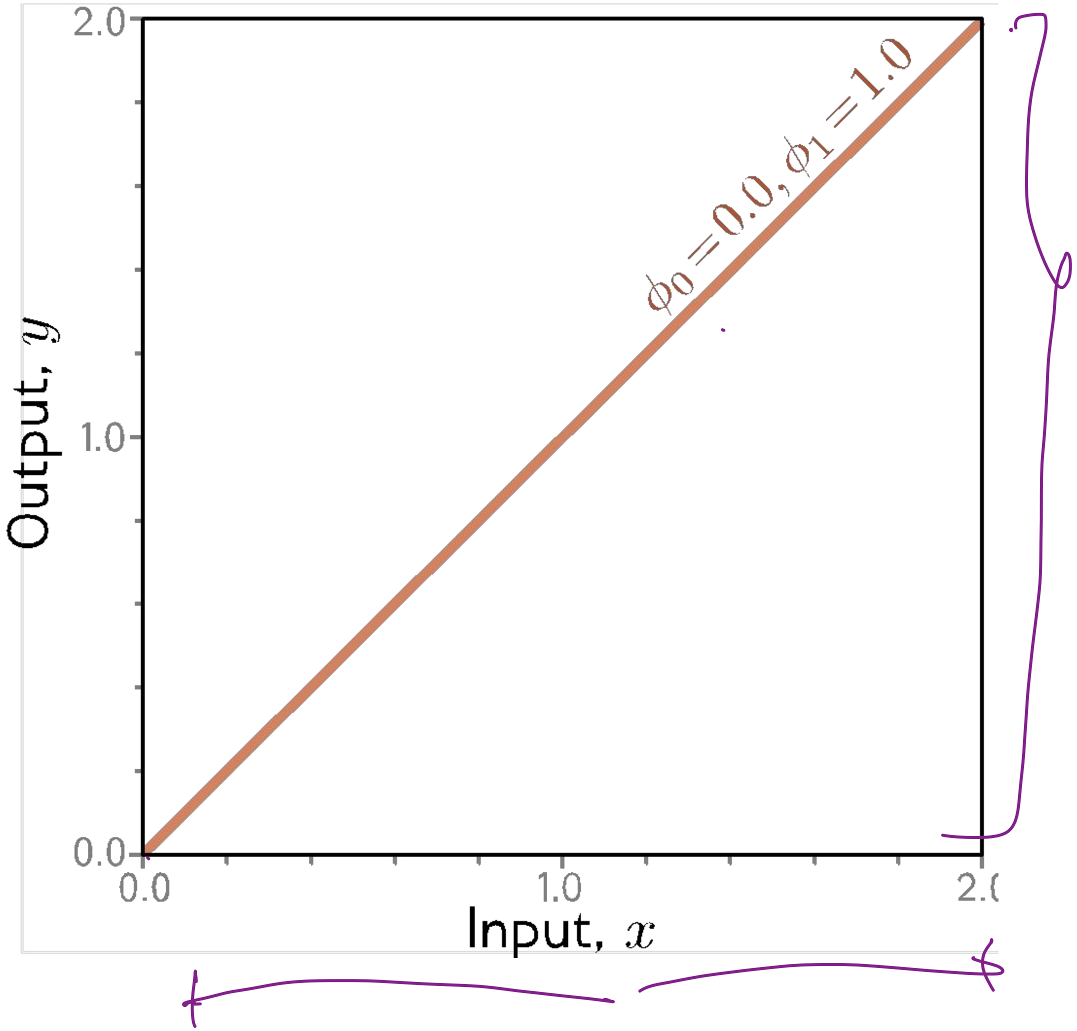
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset

← slope



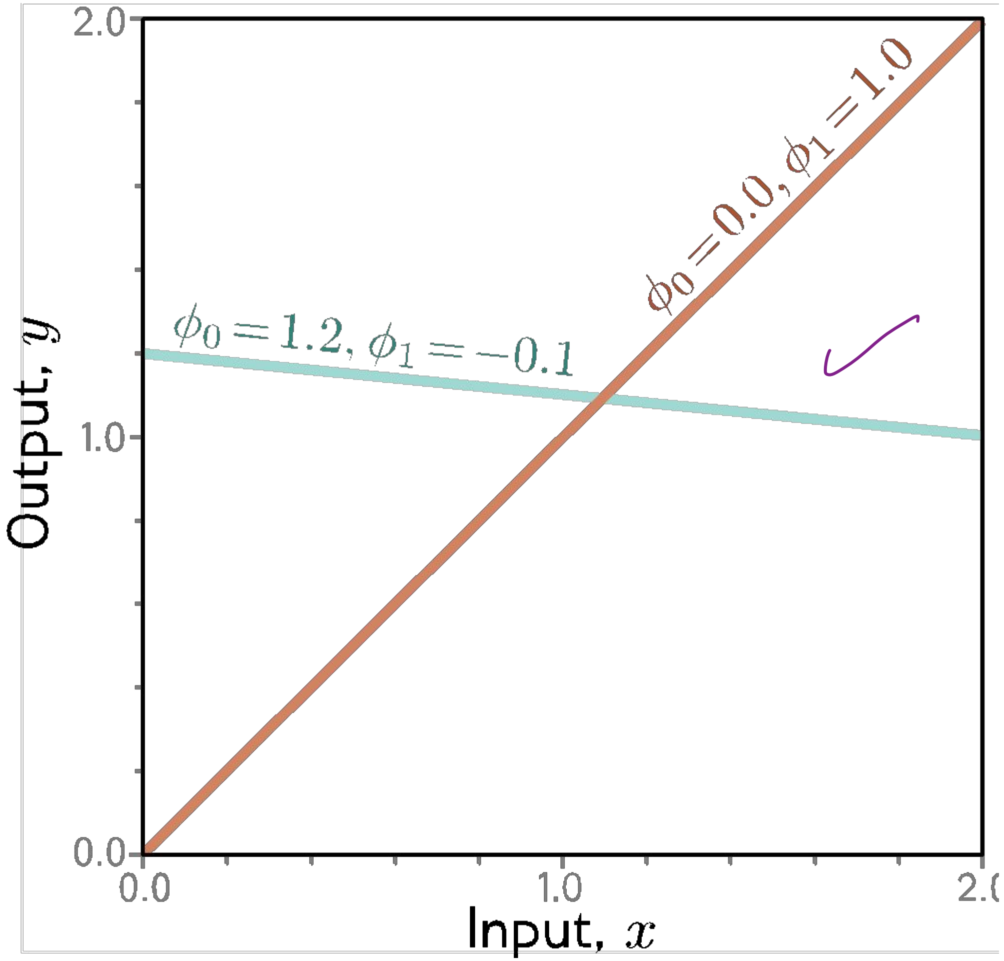
Example: 1D Linear regression model

- Model:

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{array}{l} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{array}$$



Example: 1D Linear regression model

- Model:

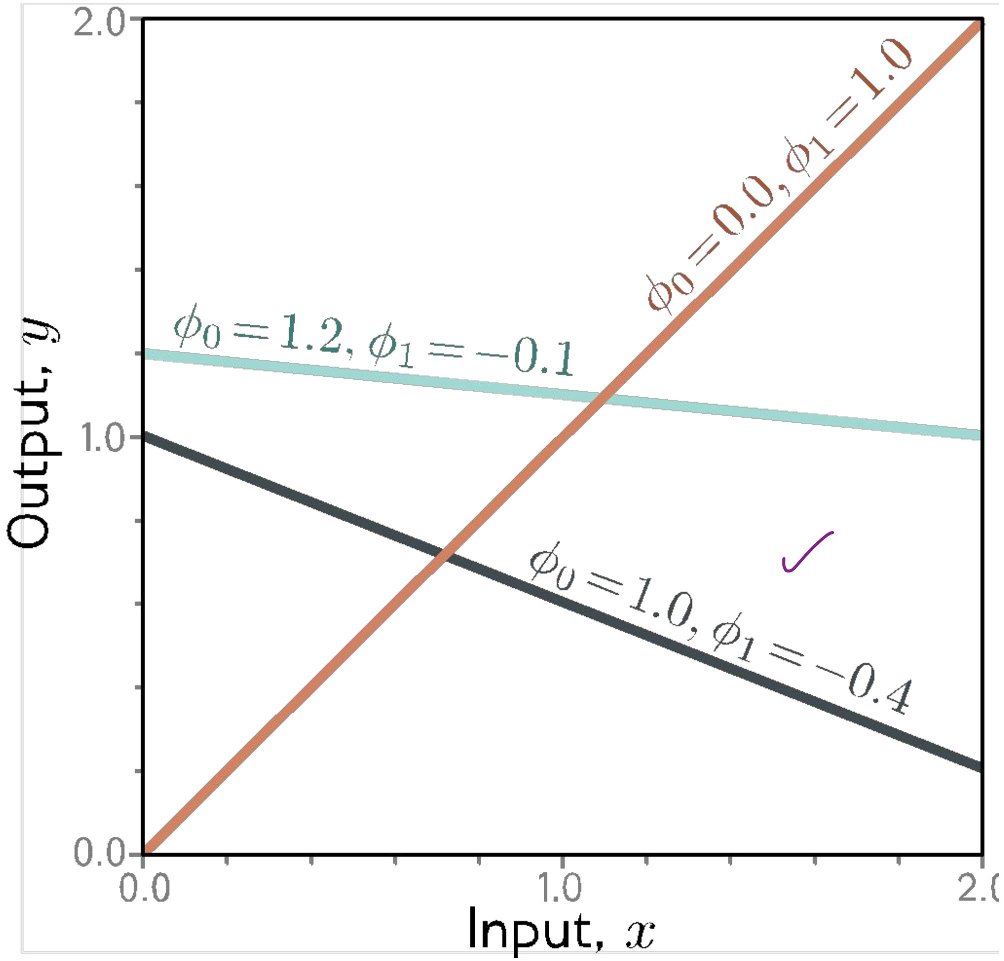
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

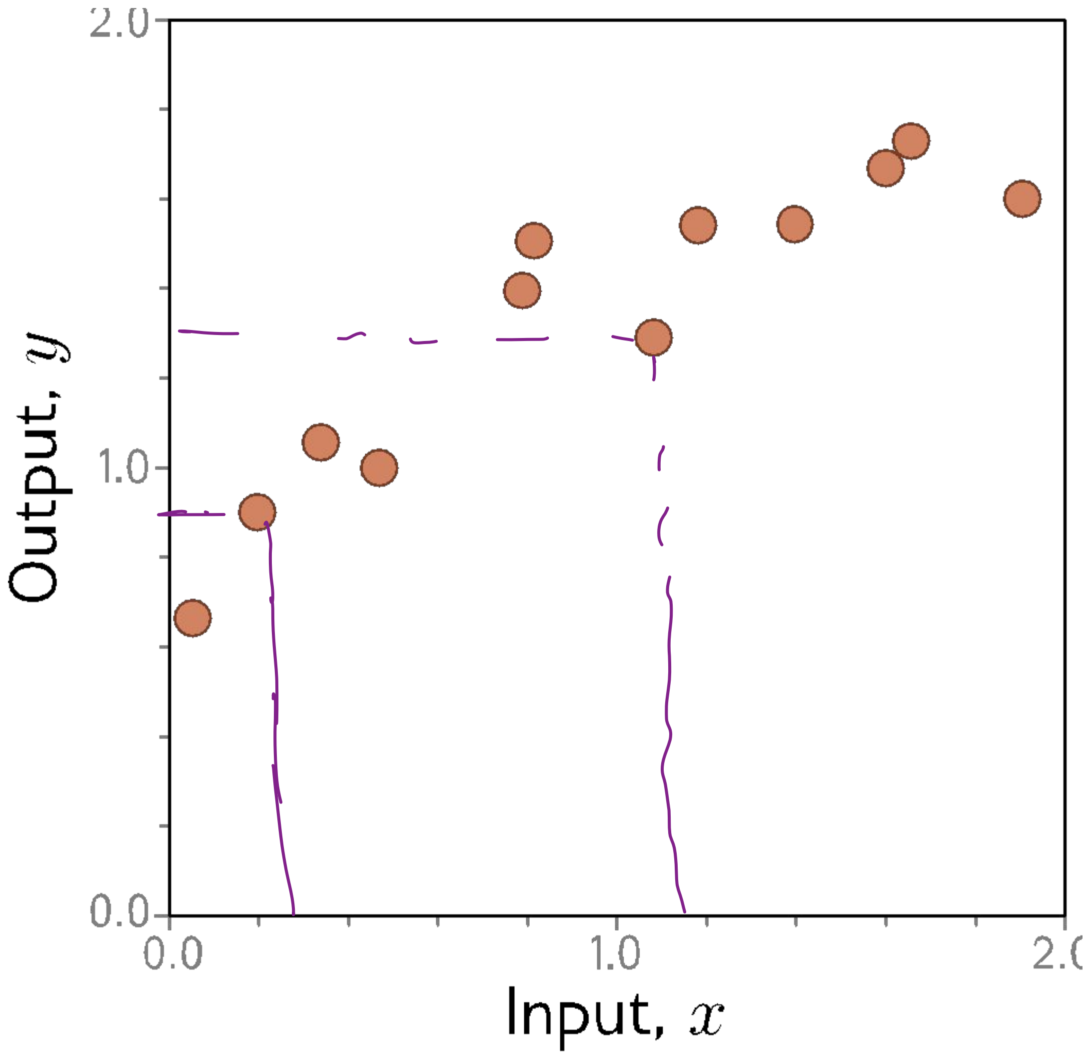
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset

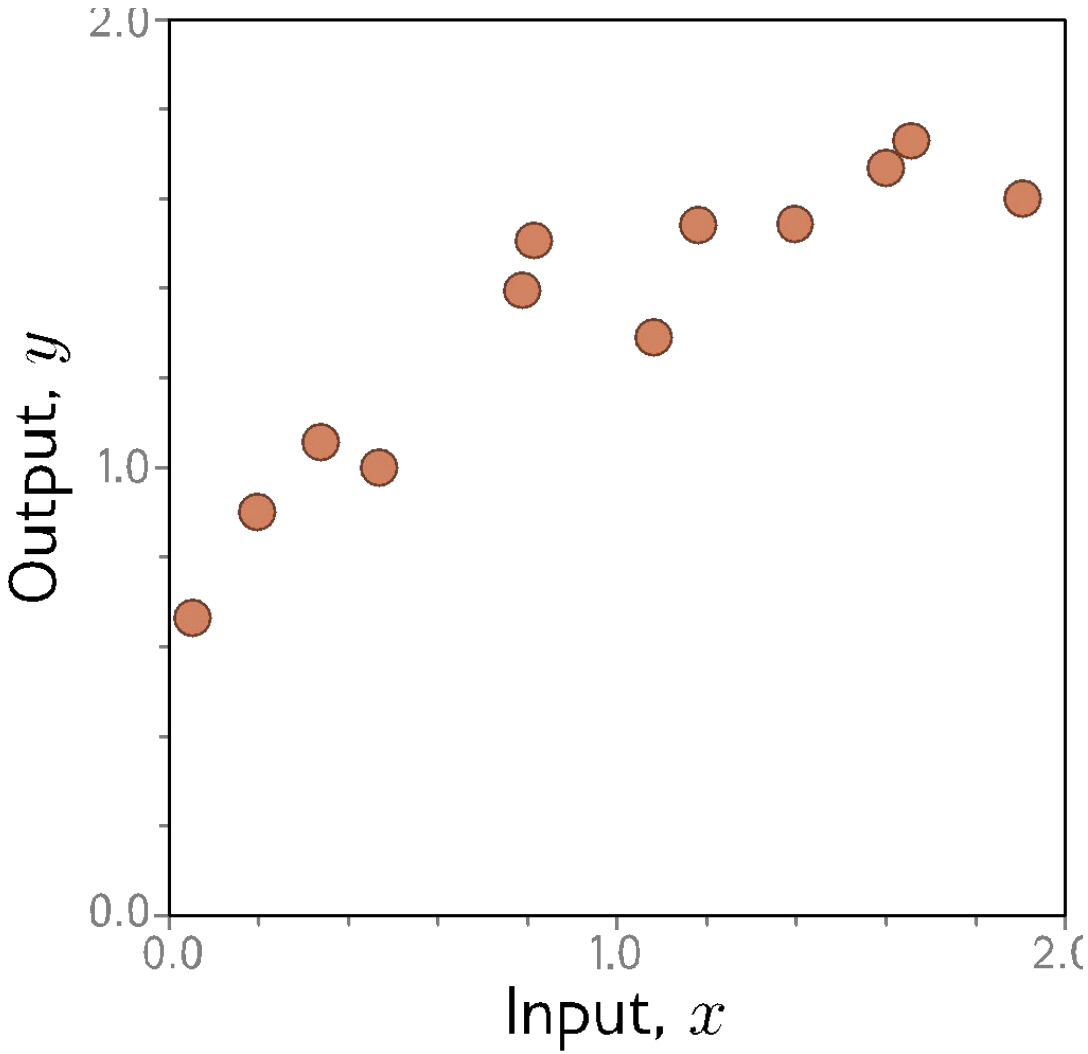
← slope



Example: 1D Linear regression training data



Example: 1D Linear regression training data

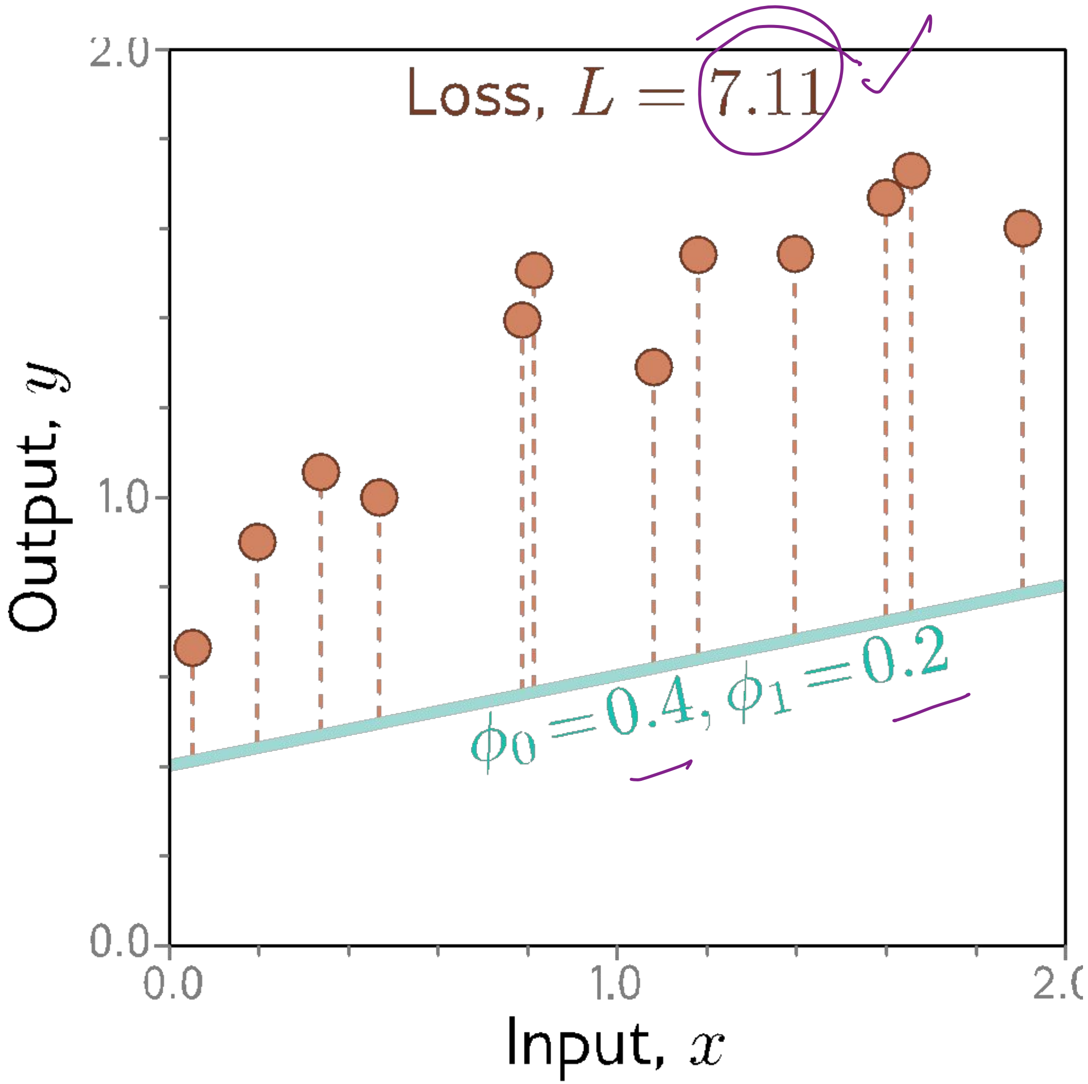


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression loss function



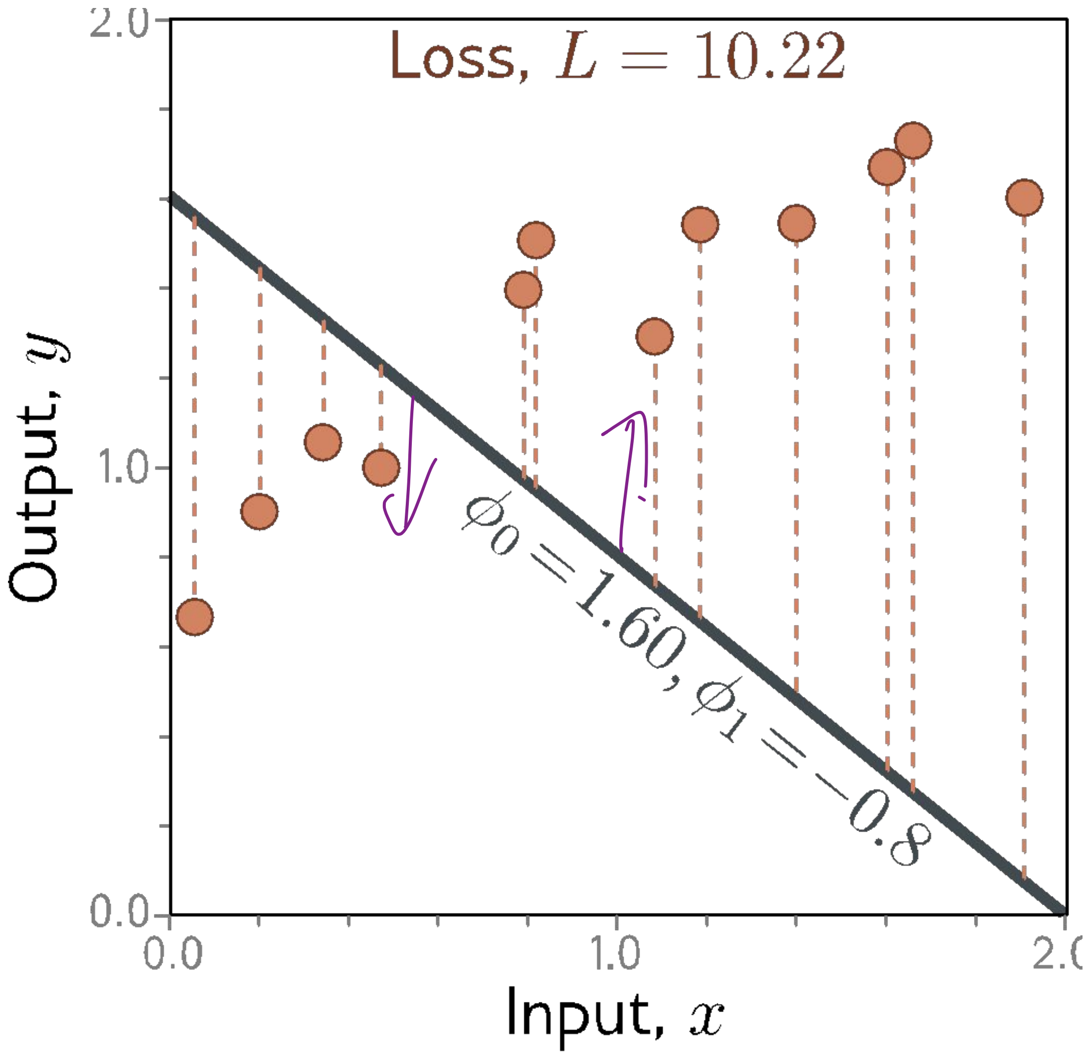
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



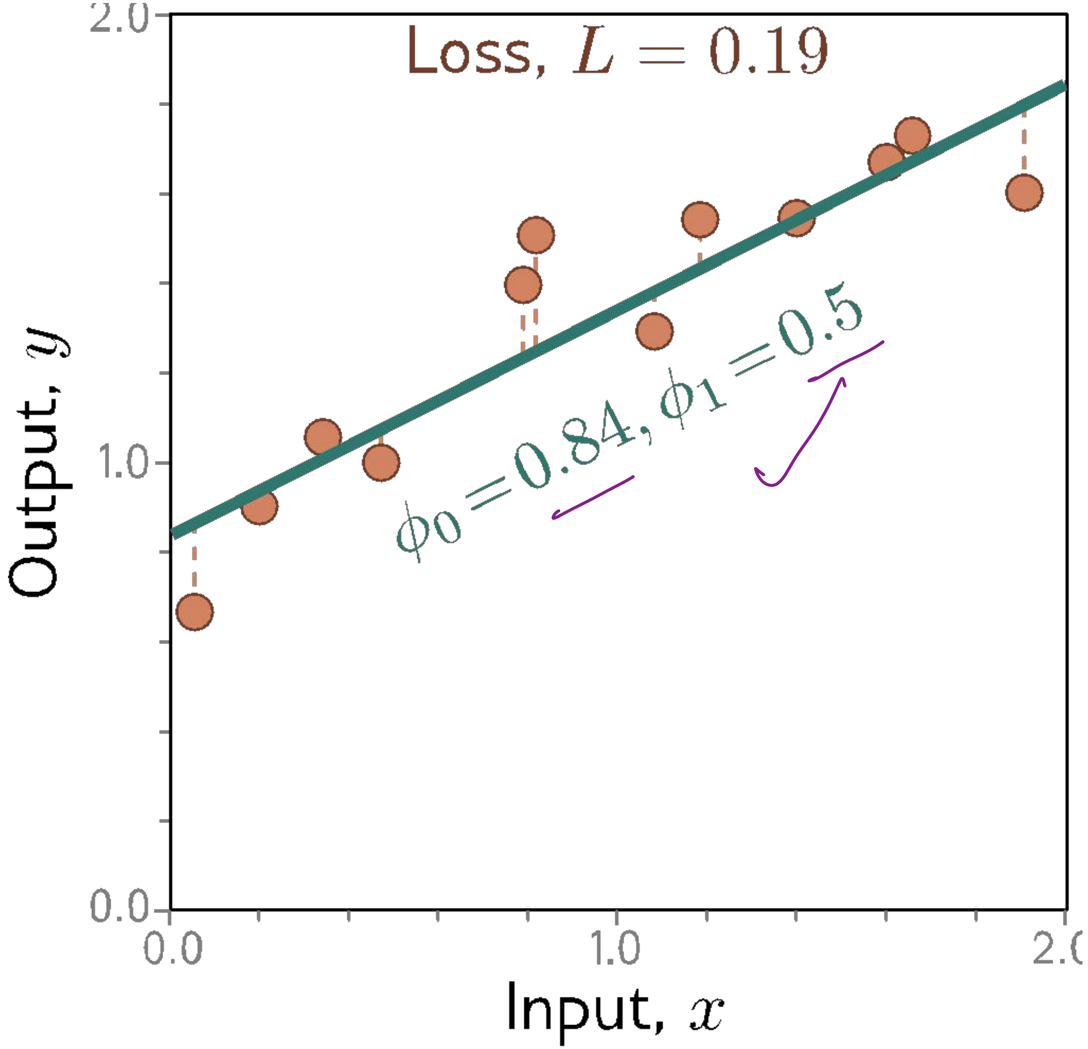
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



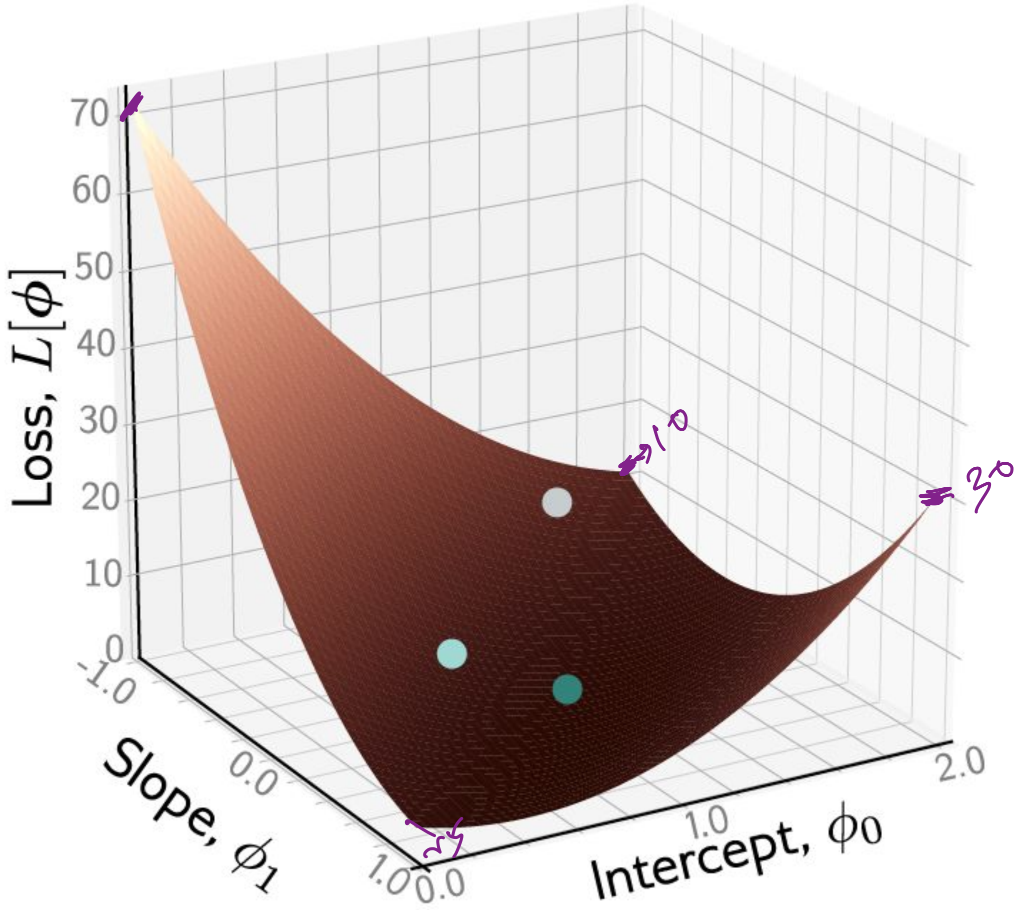
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



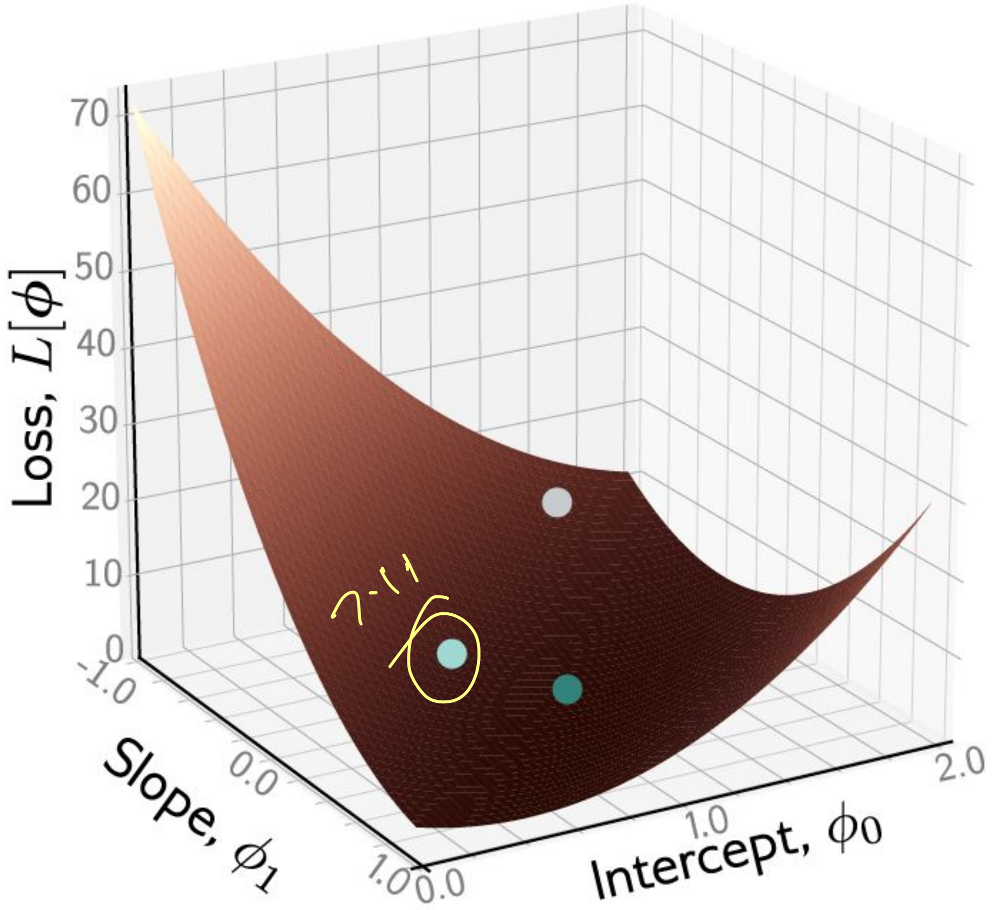
Loss function:

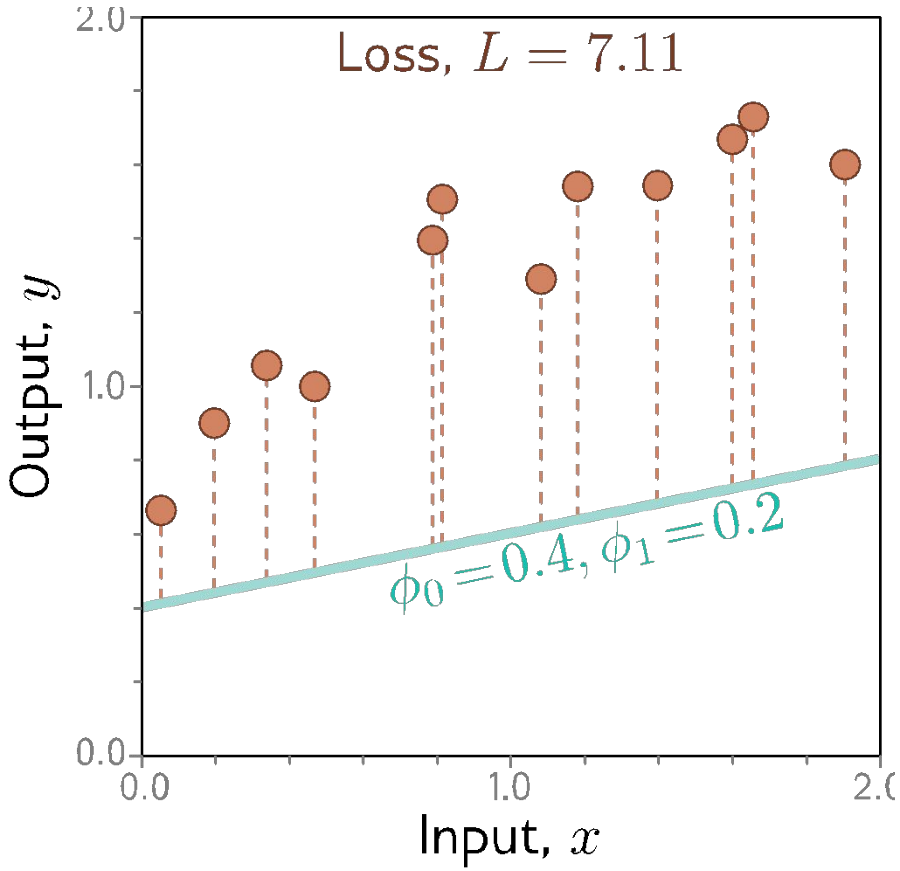
$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

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

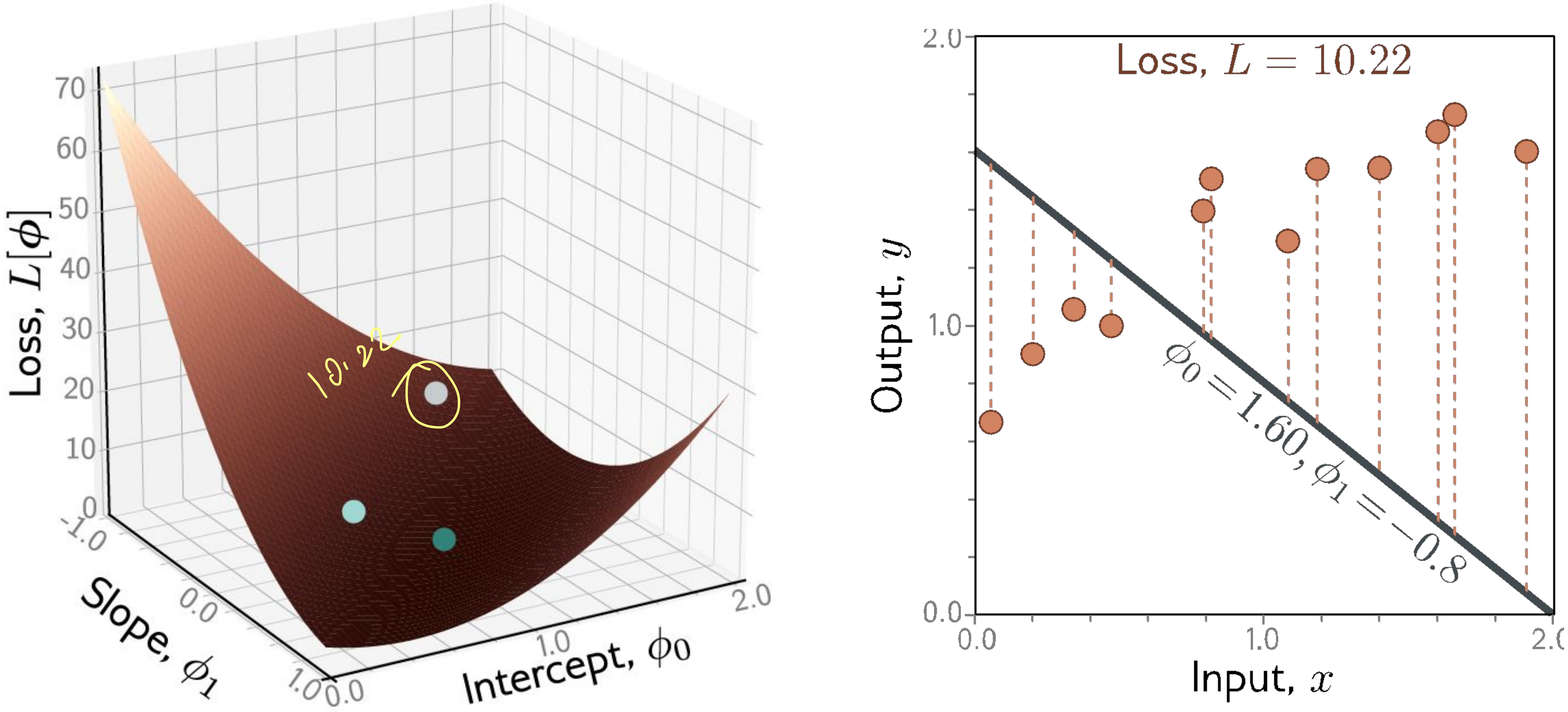
"Least squares loss function"

Example: 1D Linear regression loss function

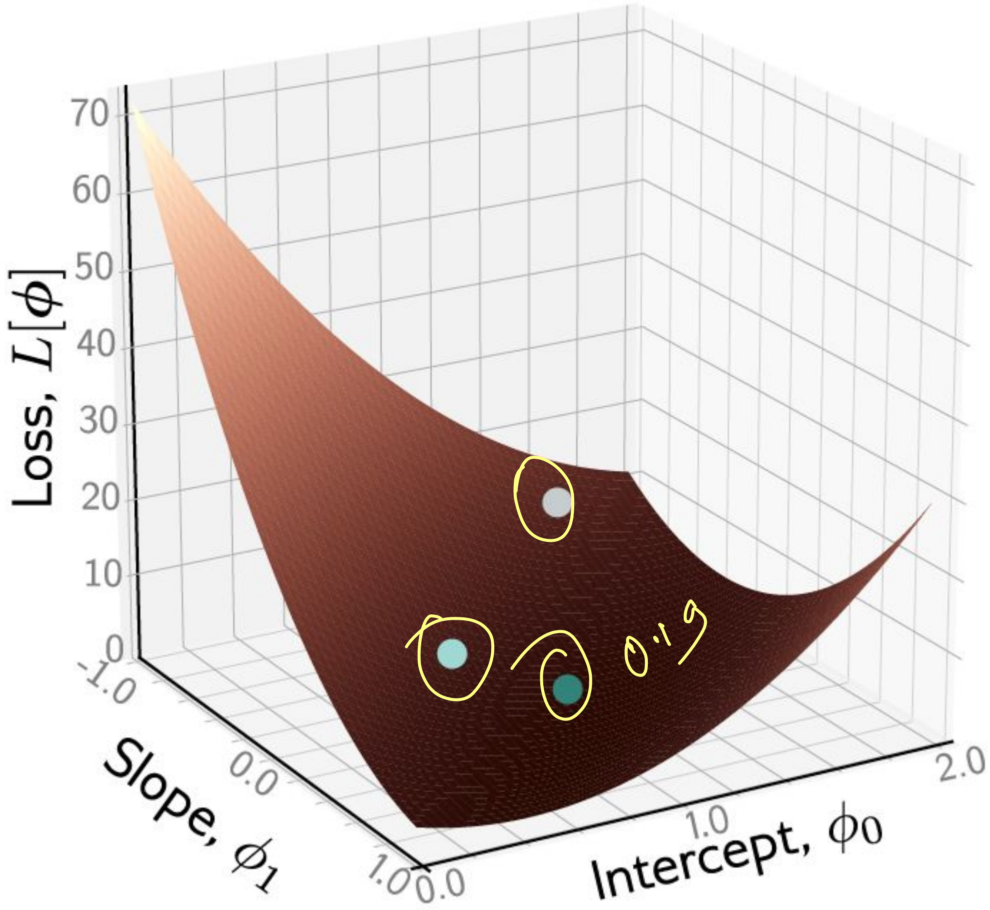


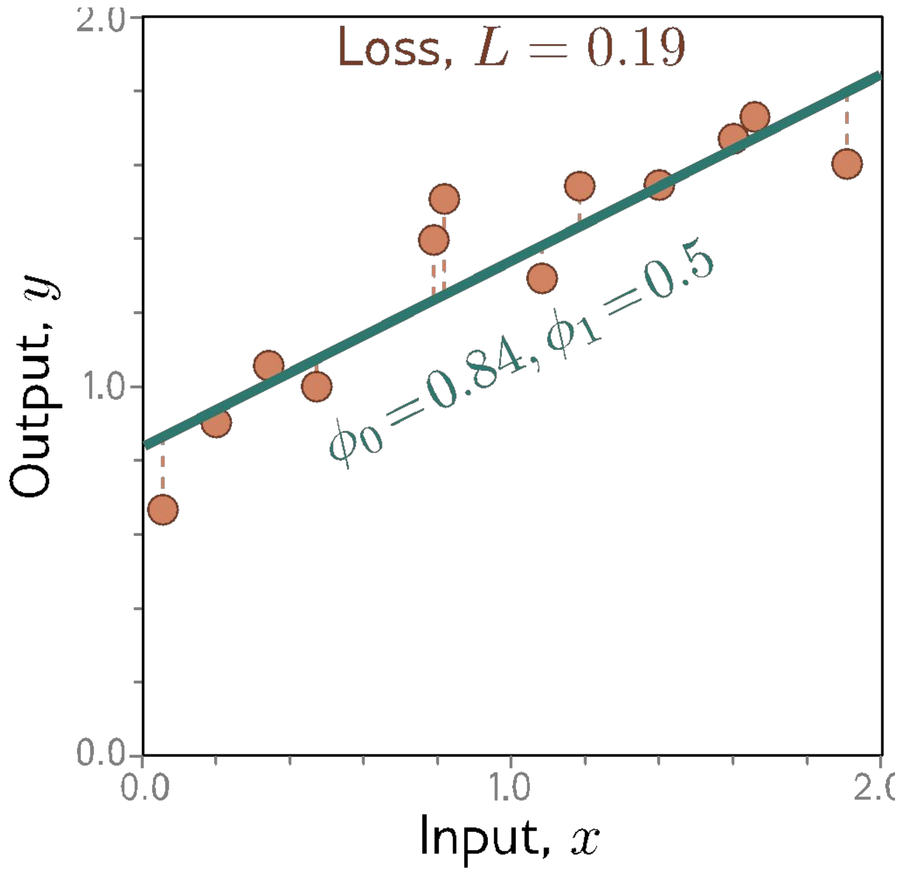


Example: 1D Linear regression loss function

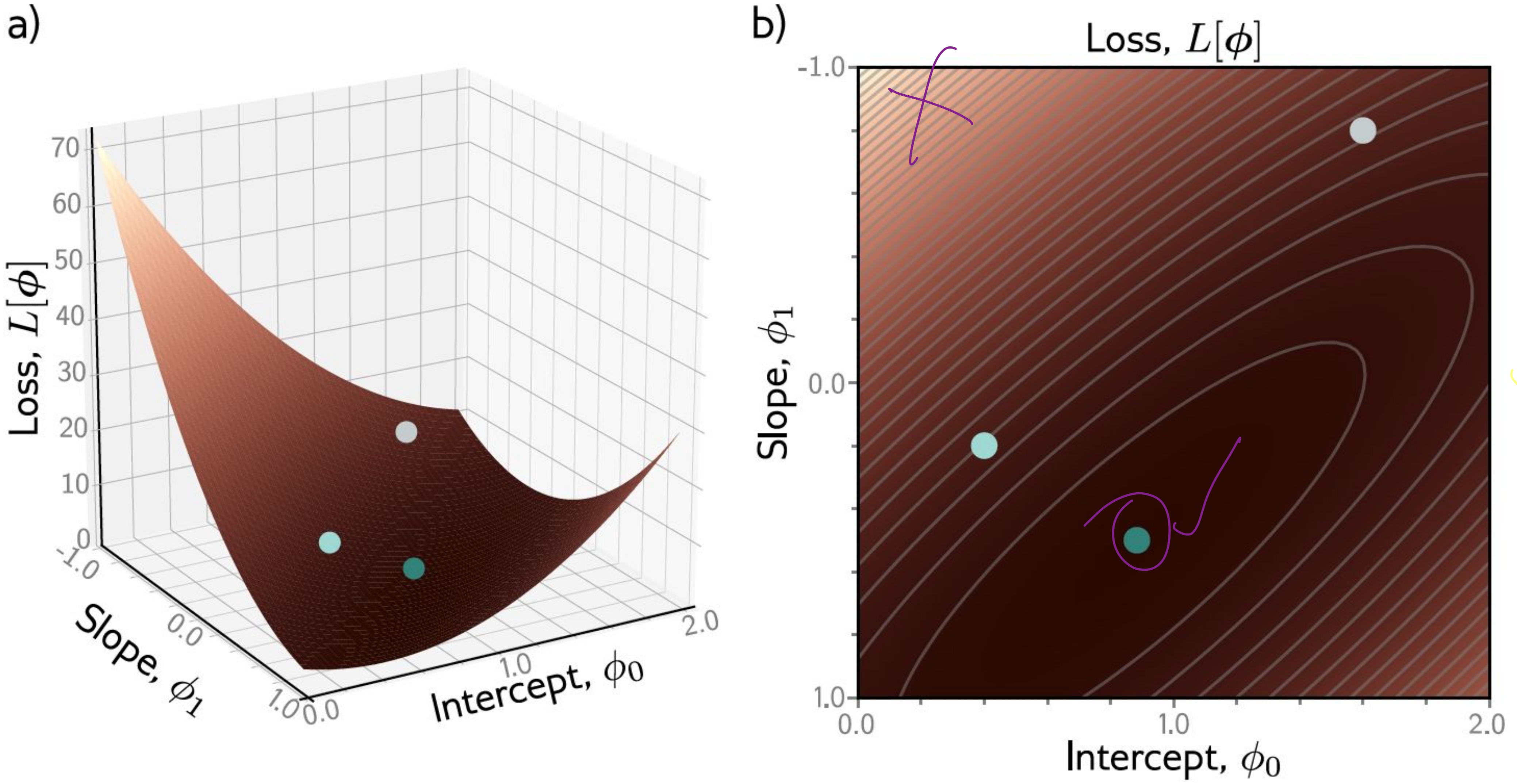


Example: 1D Linear regression loss function



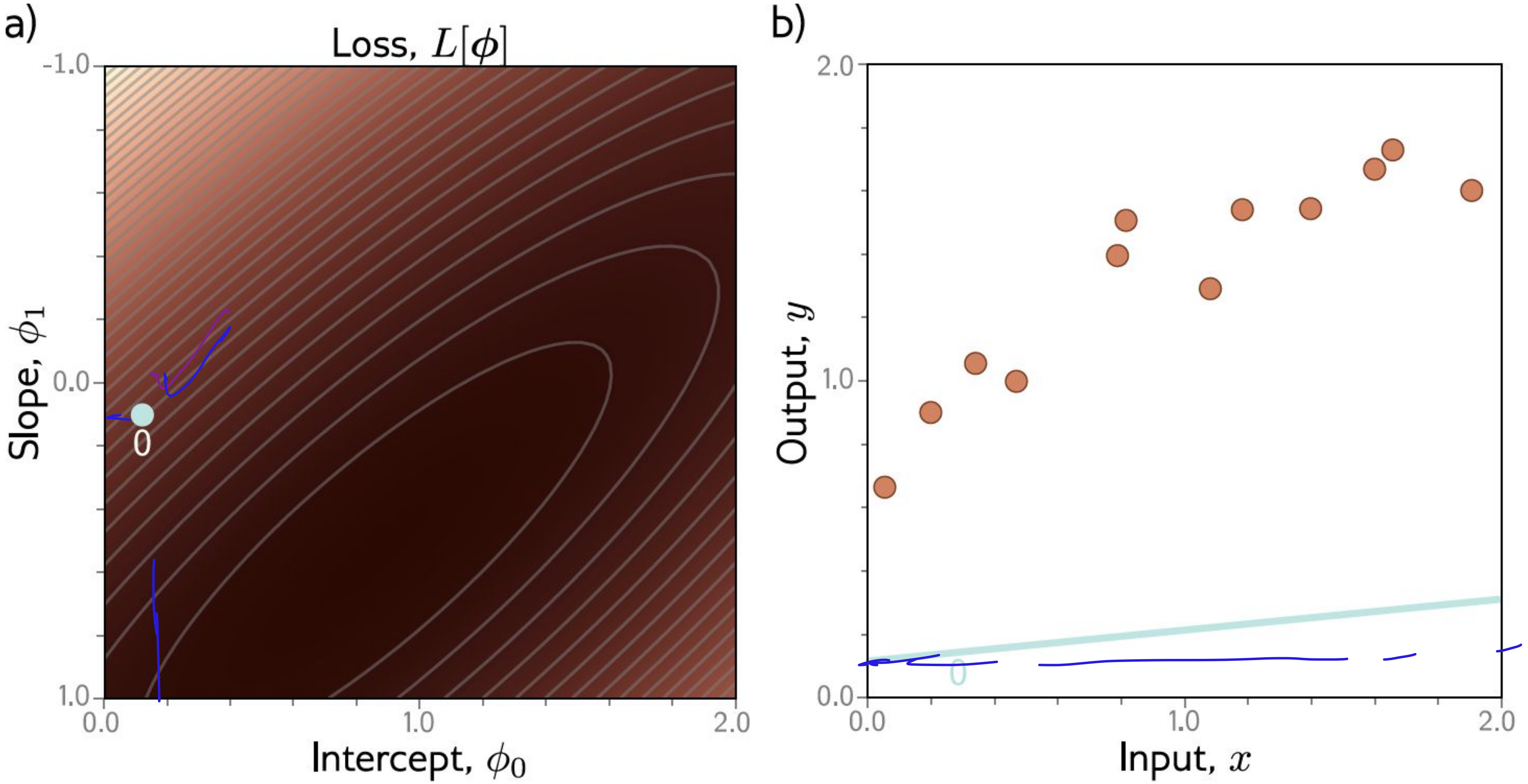


Example: 1D Linear regression loss function

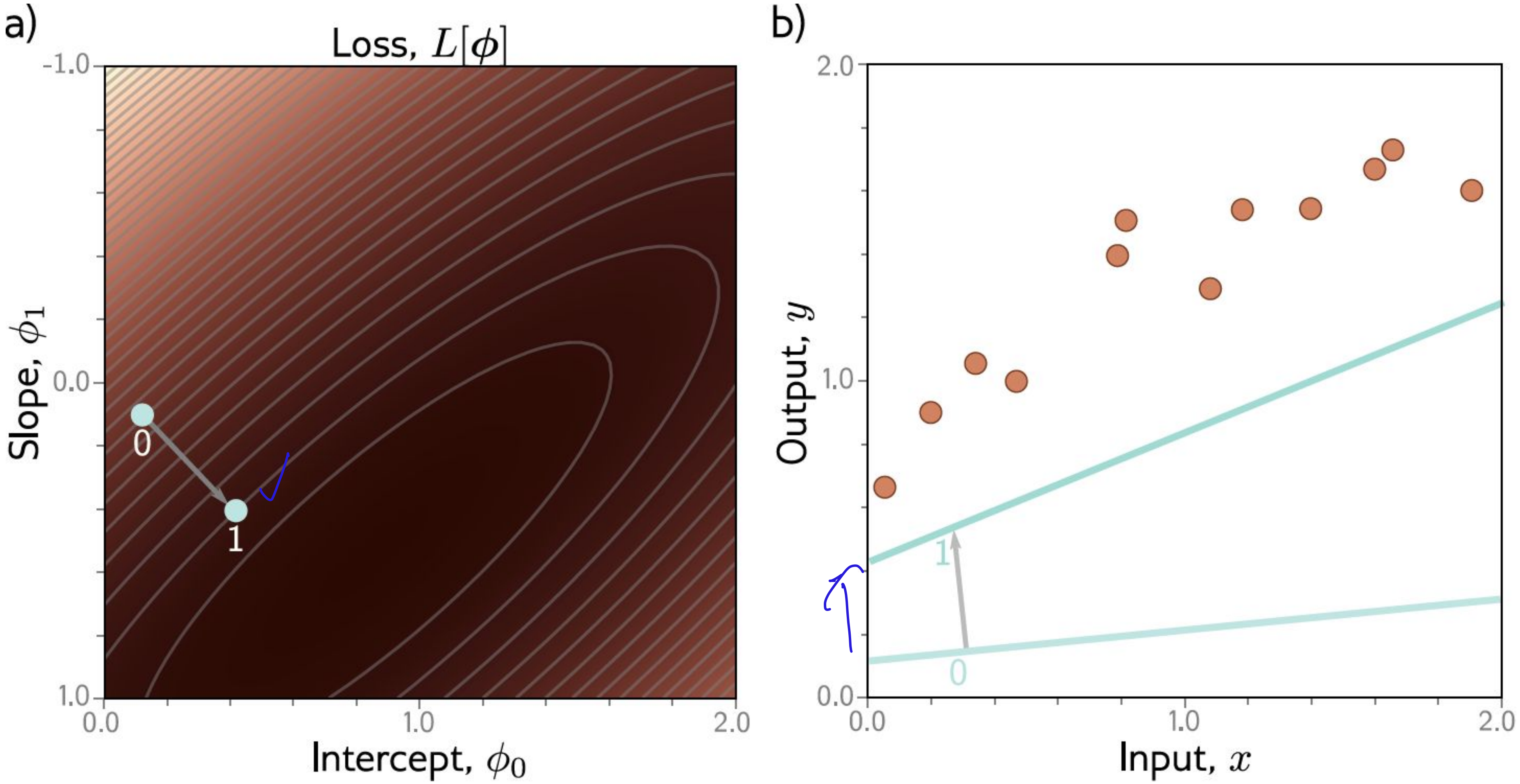


Contour

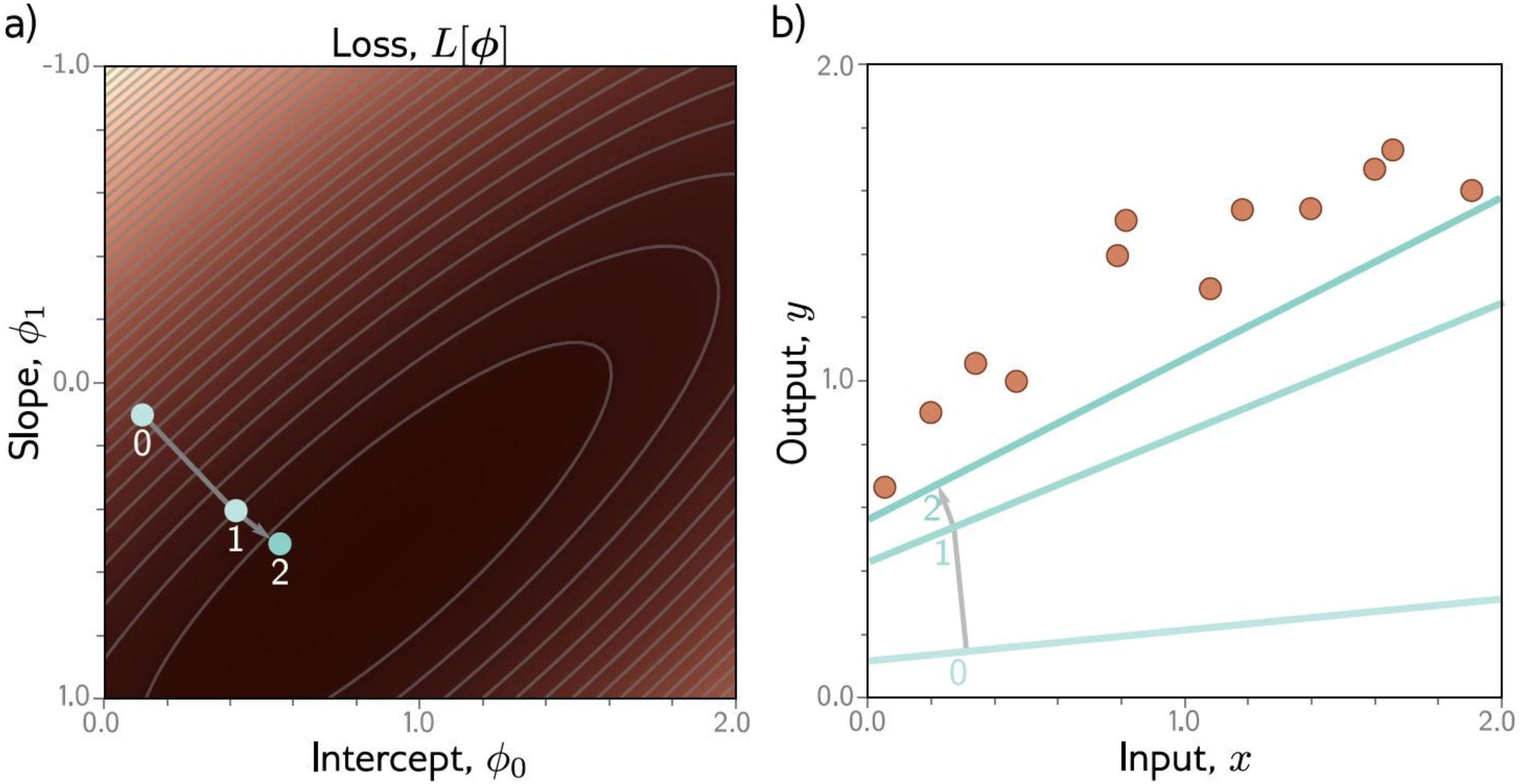
Example: 1D Linear regression training



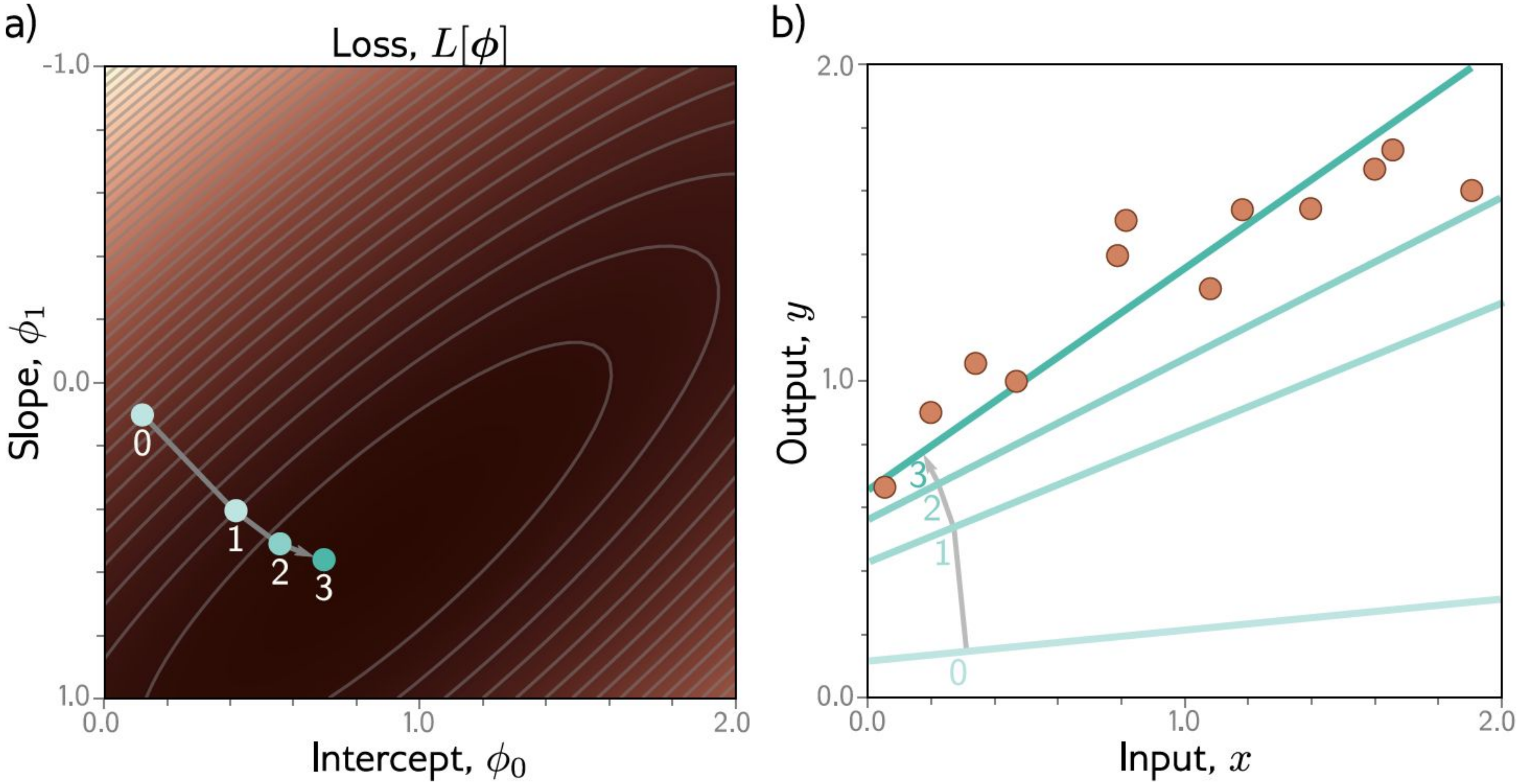
Example: 1D Linear regression training



Example: 1D Linear regression training

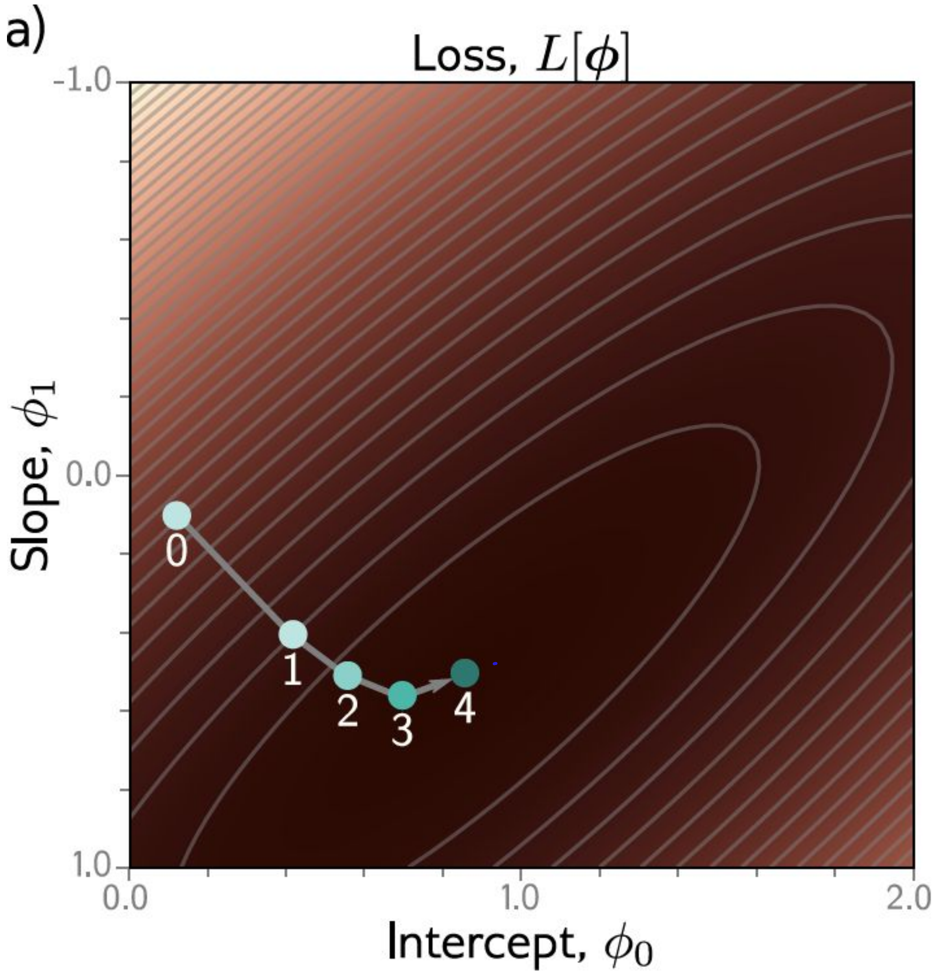


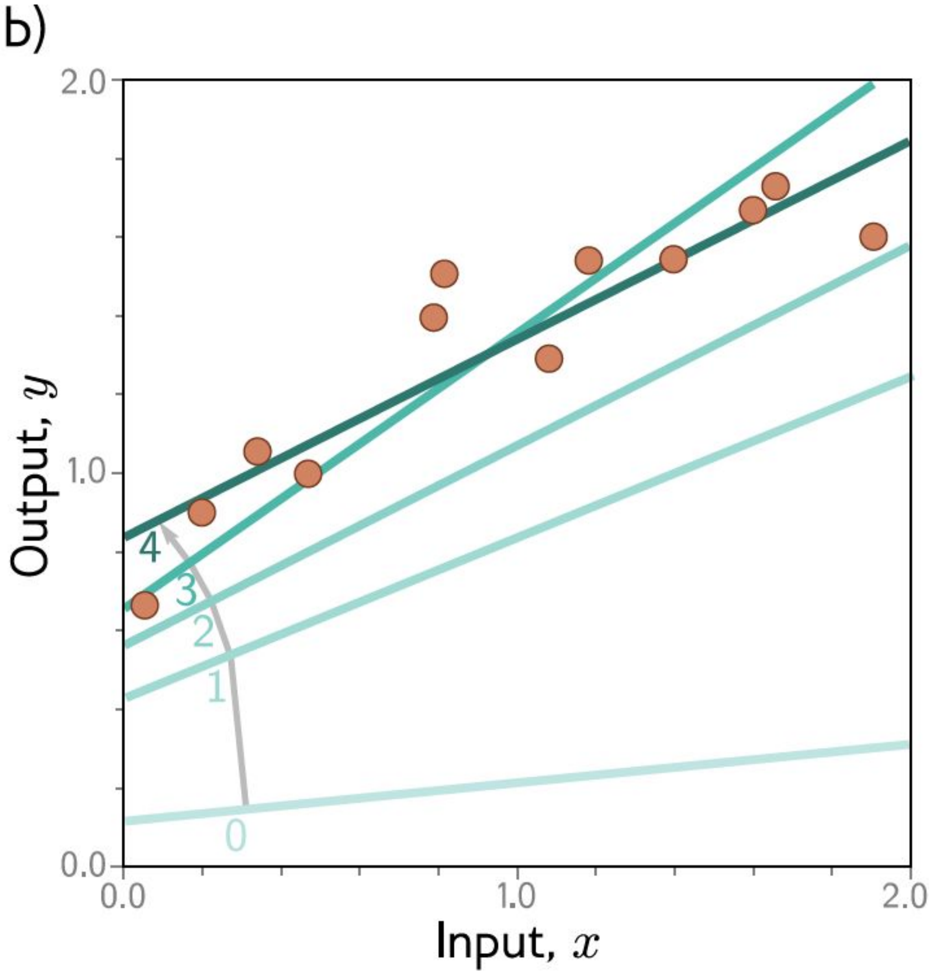
Example: 1D Linear regression training



Example: 1D Linear regression training

gradient descent





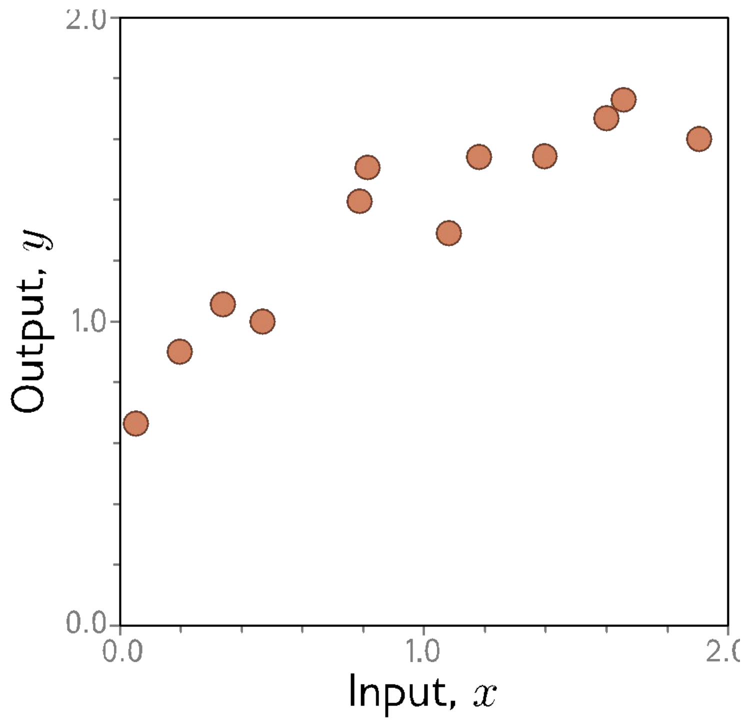
This technique is known as gradient descent

Possible objections

- But you can fit the line model in closed form!
  - Yes – but we won’t be able to do this for more complex models
- But we could exhaustively try every slope and intercept combo!
  - Yes – but we won’t be able to do this when there are millions of (~175 billion in GPT-3) parameters

Example: 1D Linear regression testing

- Test with different set of paired $\boxed{\text{input/output data}}$
  - Measure performance
  - Degree to which this is same as training = generalization
- Might not generalize well because
  - Model too simple
  - Model too complex
    - fits to statistical peculiarities of data
    - this is known as overfitting



Supervised learning

- Overview
- Notation
  - Model
  - Loss function
  - Training
  - Testing
- 1D Linear regression example
  - Model
  - Loss function
  - Training
  - Testing
- Where are we going?

Where are we going?

$\theta_0 + \theta_1 x$

- Shallow neural networks (a more flexible model)
- Deep neural networks (an even more flexible model)
- Loss functions (where did least squares come from?)
- How to train neural networks (gradient descent and variants)
- How to measure performance of neural networks (generalization)