Advanced Machine Learning

M. Ishtiaq

Supervised Learning

Supervised learning

Bicycle

Apple

Aardvark

Supervised learning

Bicycle

Apple

Aardvark

Unsupervised learning

Supervised learning

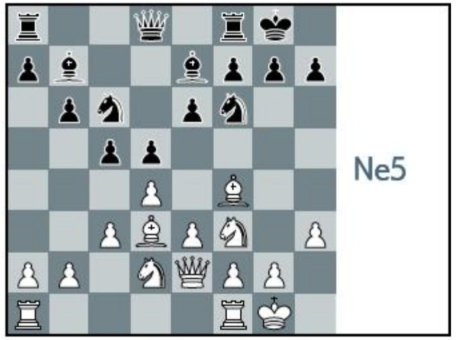
Bicycle

Apple

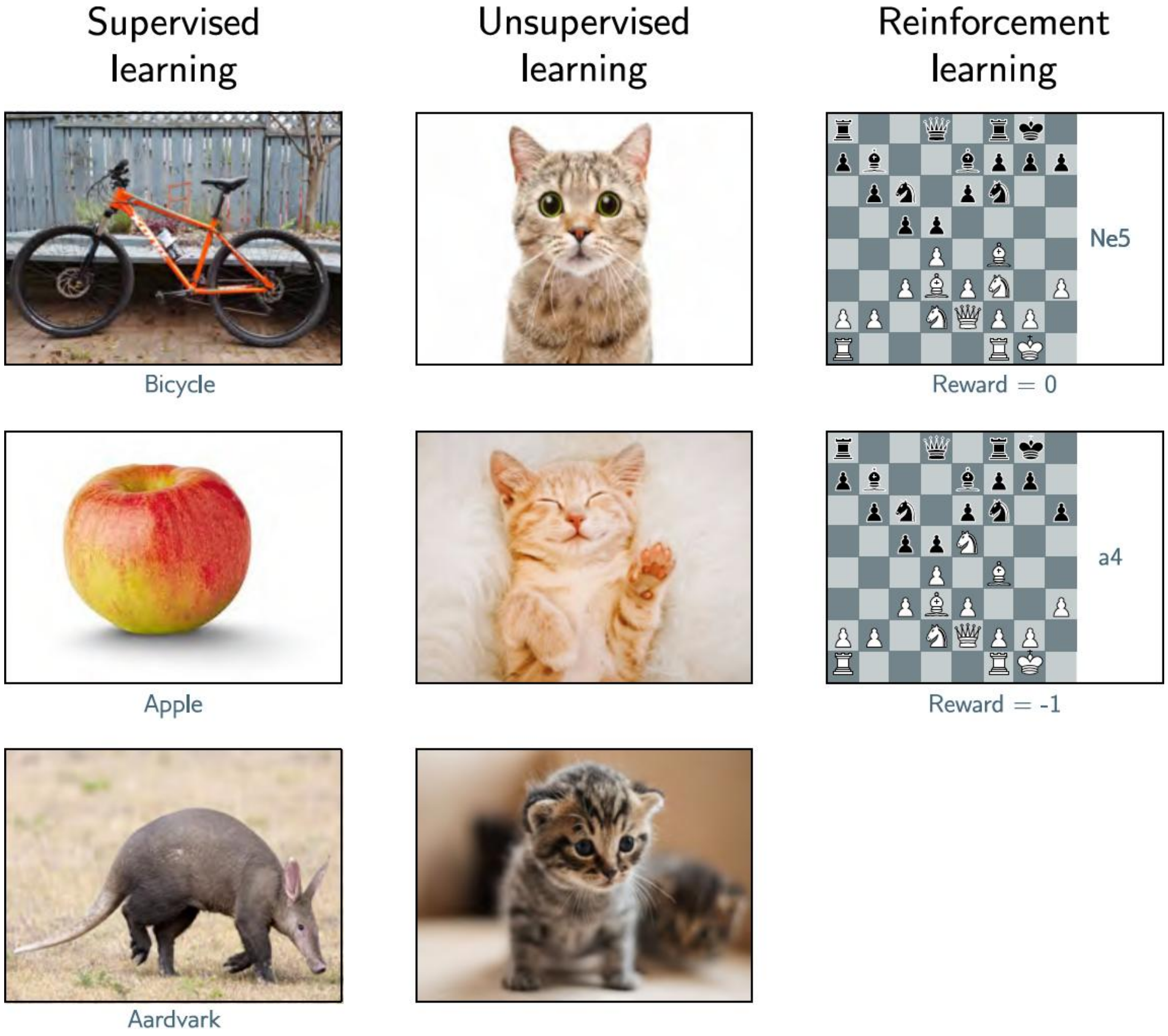
Aardvark

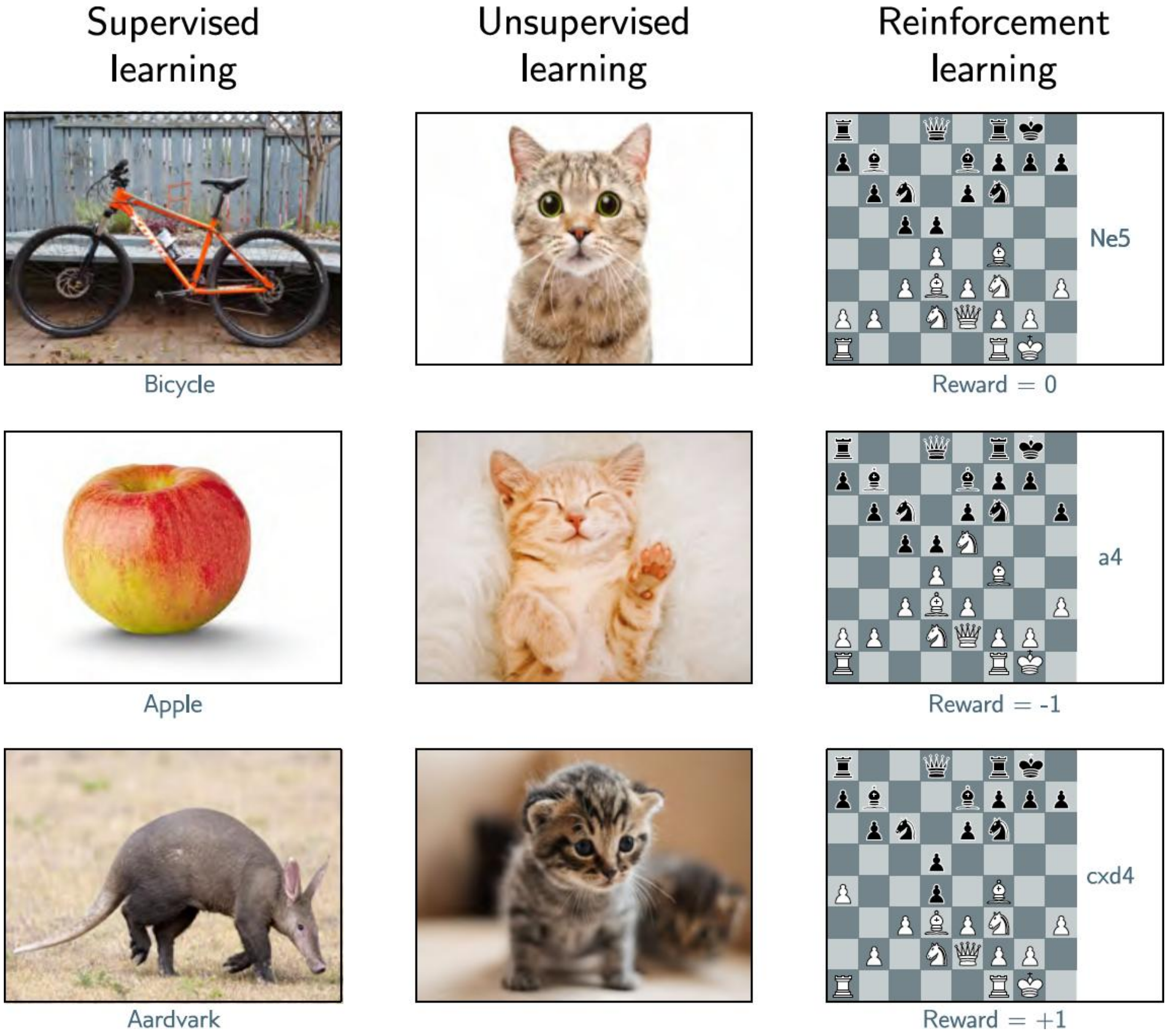
Unsupervised learning

Reinforcement learning

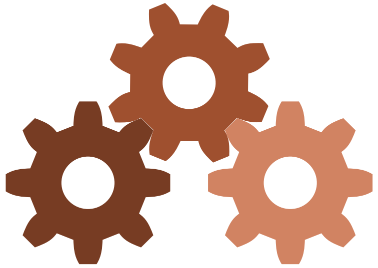


Reward $= 0$


Supervised learning
Unsupervised learning
Reinforcement learning
Bicycle
Apple
Aardvark
Ne5
Reward = 0
a4
Reward = -1


Supervised learning
Unsupervised learning
Reinforcement learning
Bicycle
Ne5
Reward = 0
Apple
a4
Reward = -1
Aardvark
cxd4
Reward = +1

Regression

Real world input	Model input	Model	Model output	Real world output
6000 square feet, 4 bedrooms, previously sold for $235K in 2005, 1 parking spot.	$\begin{bmatrix} 6000 \\ 4 \\ 235 \\ 2005 \\ 1 \end{bmatrix}$	 Supervised learning model	$[340]$	Predicted price is $340k

- Univariate regression problem (one output, real value)

Supervised learning

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

Problems

Problem 3.1 What kind of mapping from input to output would be created if the activation function in equation 3.1 was linear so that $a[z] = \psi_0 + \psi_1 z$? What kind of mapping would be created if the activation function was removed, so $a[z] = z$?

Problem 3.2 For each of the four linear regions in figure 3.3j, indicate which hidden units are inactive and which are active (i.e., which do and do not clip their inputs).

Problem 3.3* Derive expressions for the positions of the "joints" in function in figure 3.3j in terms of the ten parameters $\boldsymbol{\phi}$ and the input x. Derive expressions for the slopes of the four linear regions.

Problem 3.4 Draw a version of figure 3.3 where the y-intercept and slope of the third hidden unit have changed as in figure 3.14c. Assume that the remaining parameters remain the same.

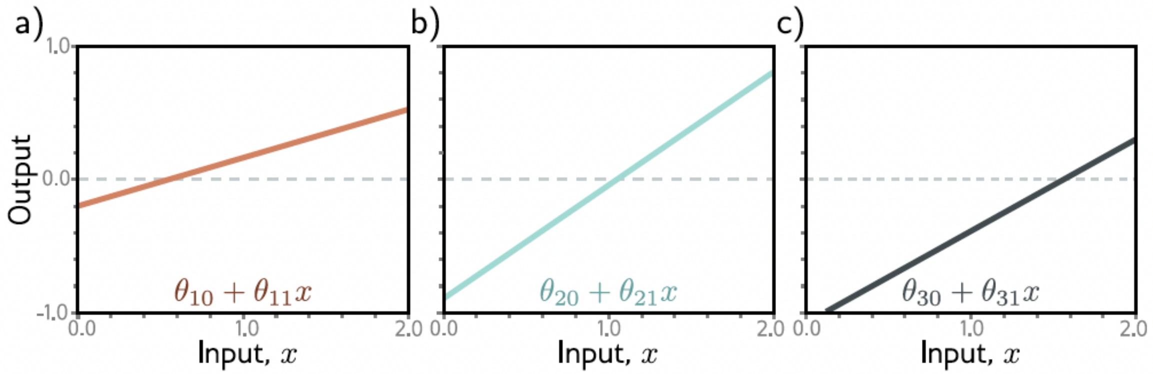


Figure 3.14 Processing in network with one input, three hidden units, and one output for problem 3.4. a–c) The input to each hidden unit is a linear function of the inputs. The first two are the same as in figure 3.3, but the last one differs.

Problem 3.5 Prove that the following property holds for $\alpha \in \mathbb{R}^+$:

$$\mathrm{ReLU}[\alpha \cdot z] = \alpha \cdot \mathrm{ReLU}[z]. \tag{3.14}$$

This is known as the *non-negative homogeneity* property of the ReLU function.

Supervised learning

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

Supervised learning overview

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

Supervised learning overview

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

- Computing the inputs from the outputs = inference

- Example:
  - Input is age and milage of secondhand 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

- Computing the inputs from the outputs = inference

Supervised learning overview

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

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

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

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

- 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
- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a family of equations

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Supervised learning overview

- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a family of equations

- Computing the inputs from the outputs = 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

Supervised learning

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

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:

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

- Model:

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

Model

- Parameters:

$$\phi$$

Parameters always Greek letters

- Model :

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

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:

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

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:

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

or for short:

$$L[\phi]$$

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:

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

Testing

- To test the model, run on a separate test dataset of input / output pairs
- See how well it generalizes to new data

Supervised learning

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

Example: 1D Linear regression model

- Model:

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

- 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:

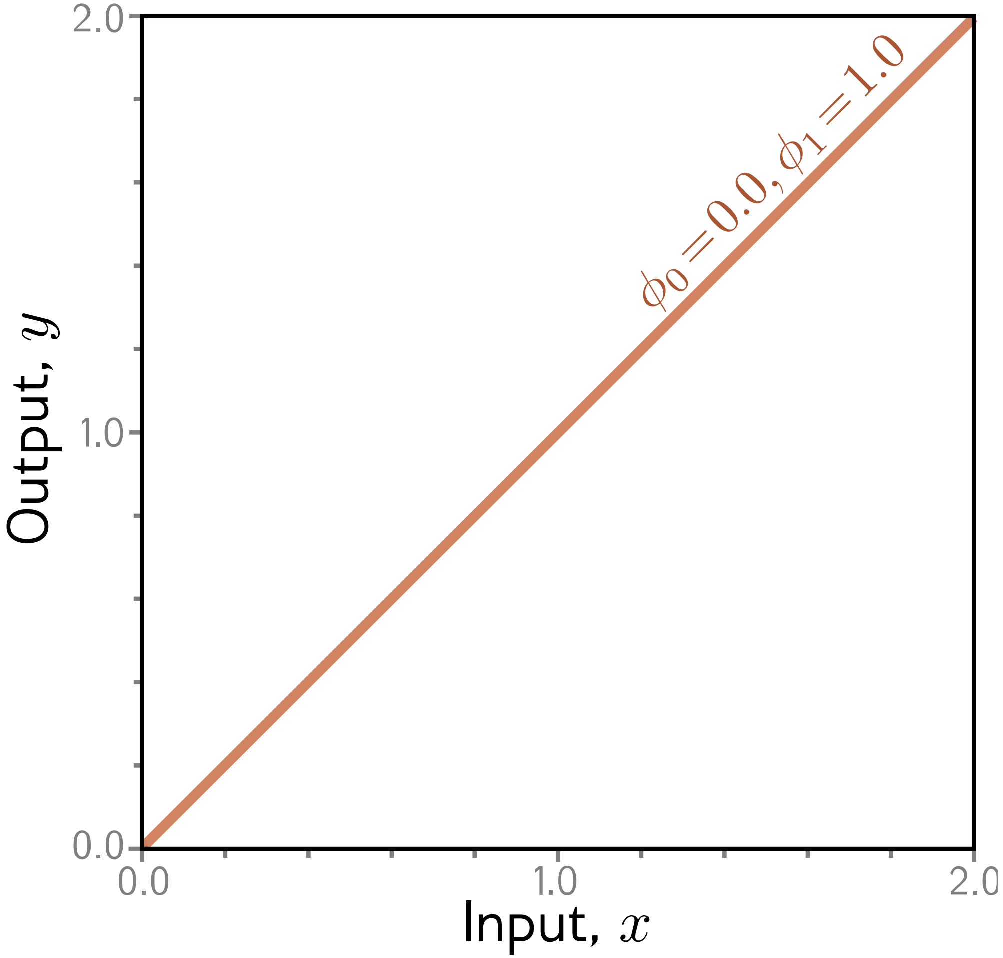
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

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

← y-offset

← slope



Example: 1D Linear regression model

- Model:

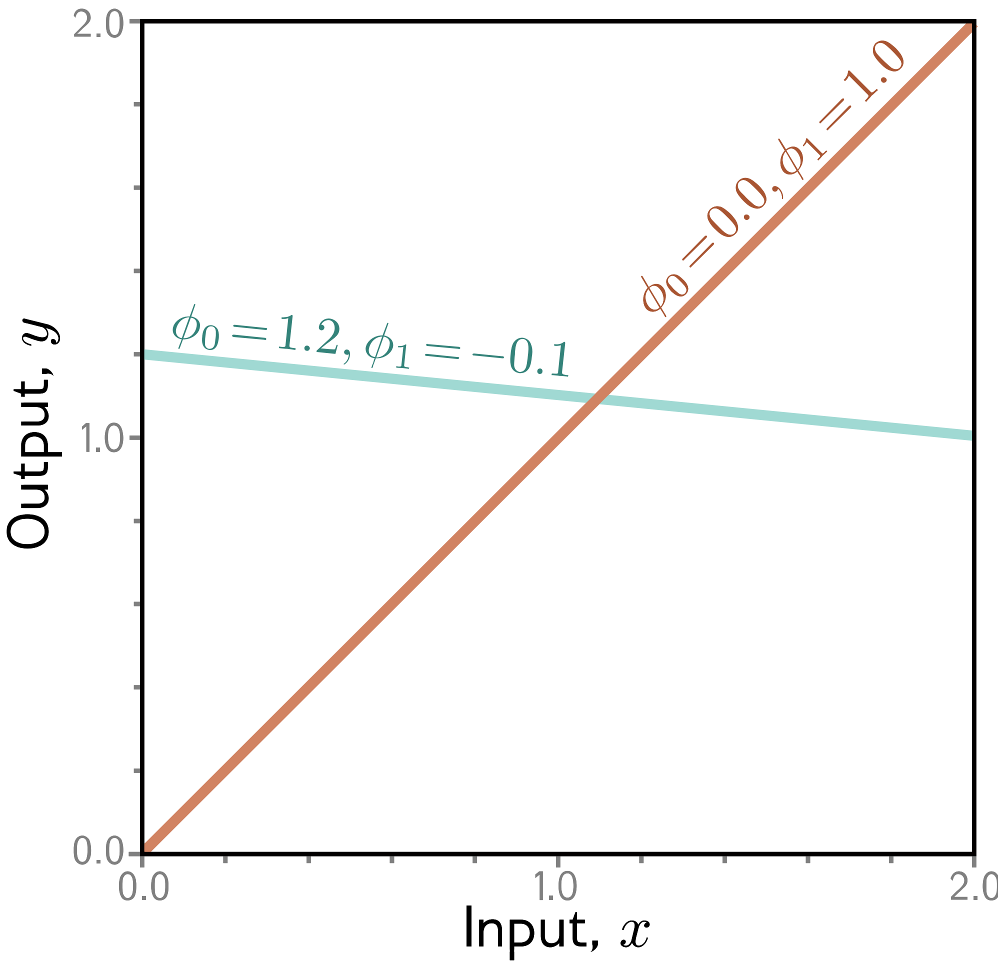
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

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

y-offset

slope



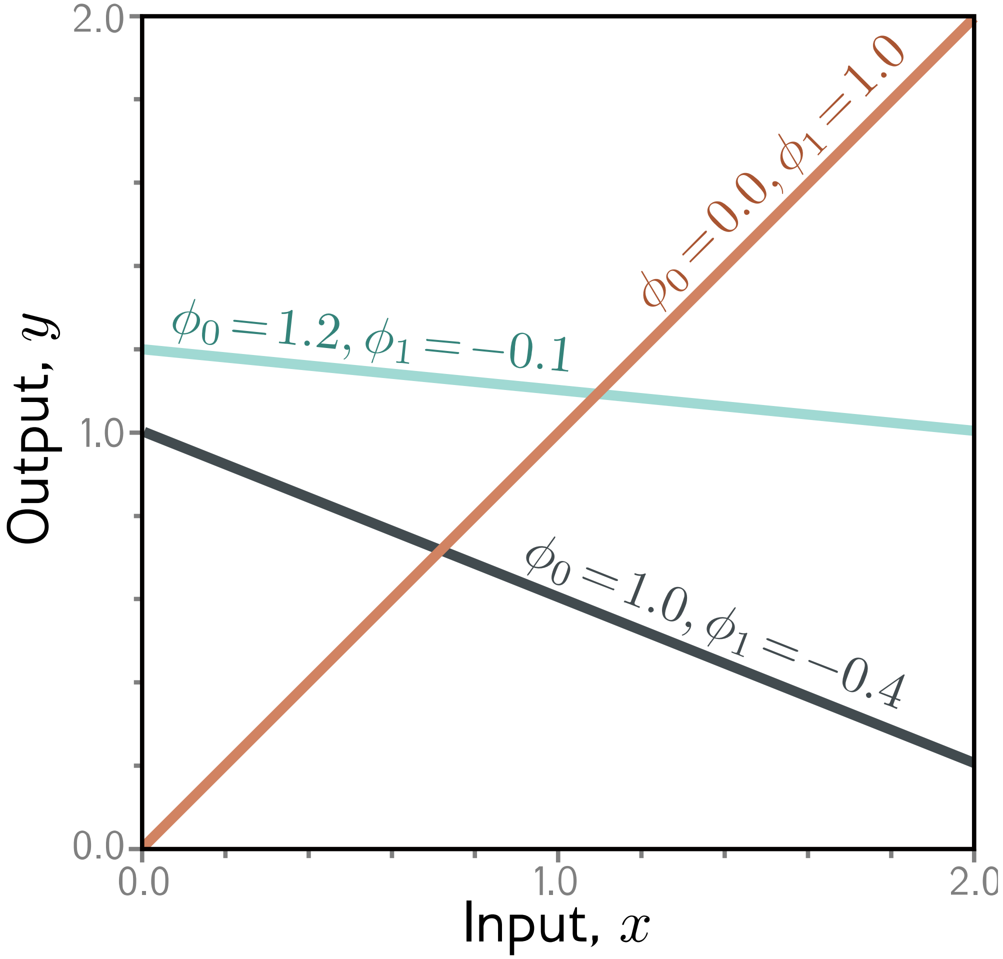
Example: 1D Linear regression model

- Model:

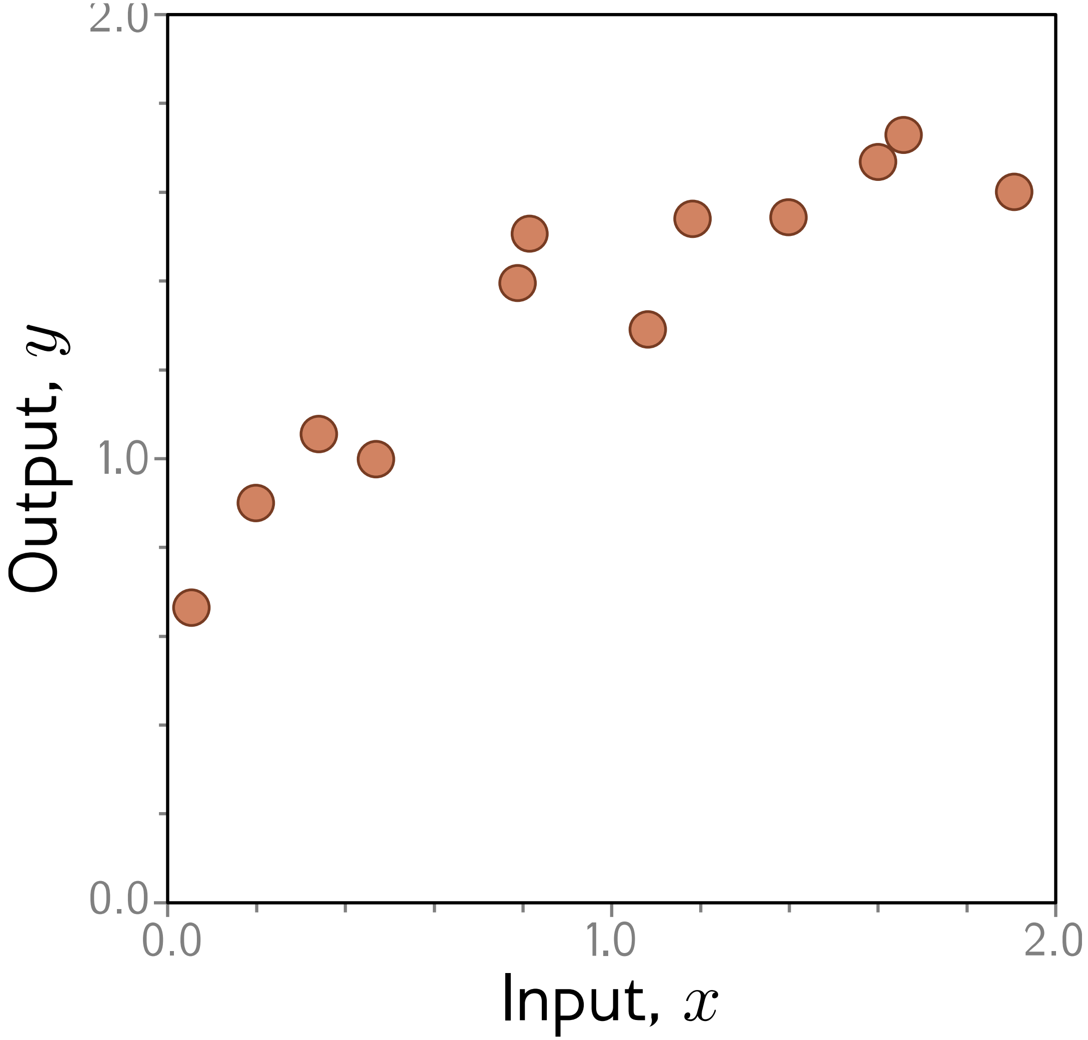
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

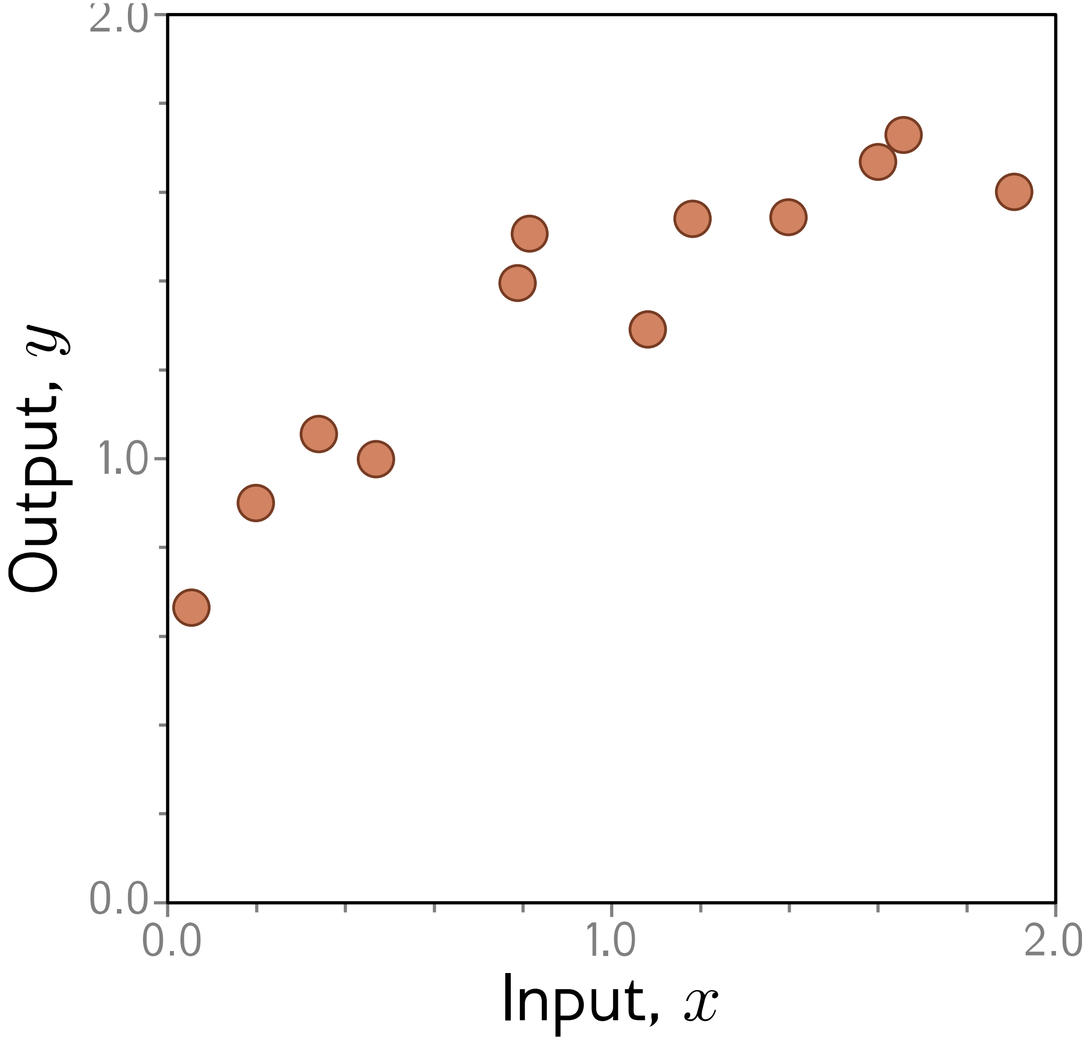
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix} \begin{array}{l} \leftarrow \text{y-offset} \\ \leftarrow \text{slope} \end{array}$$



Example: 1D Linear regression training data



Example: 1D Linear regression training data

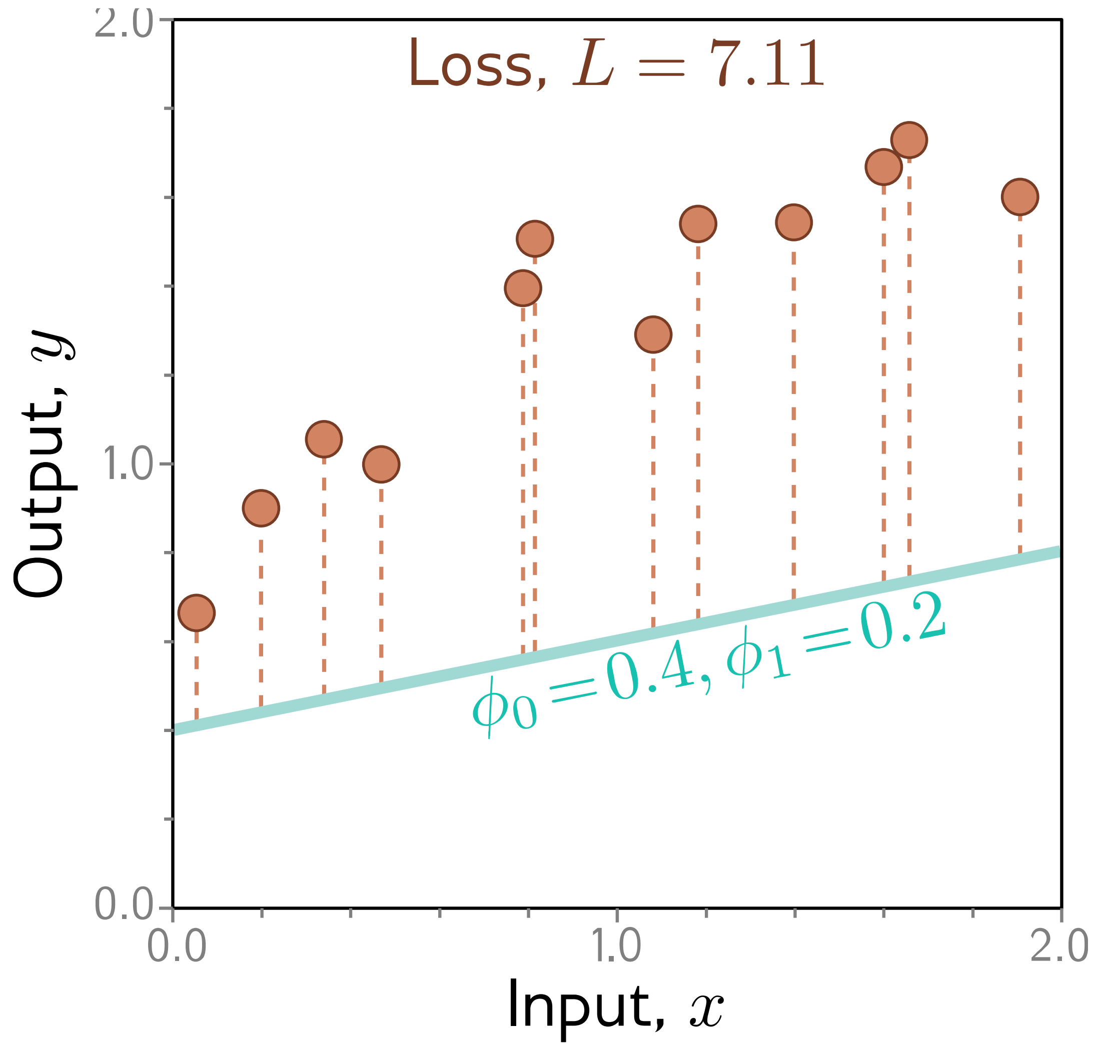


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression loss function



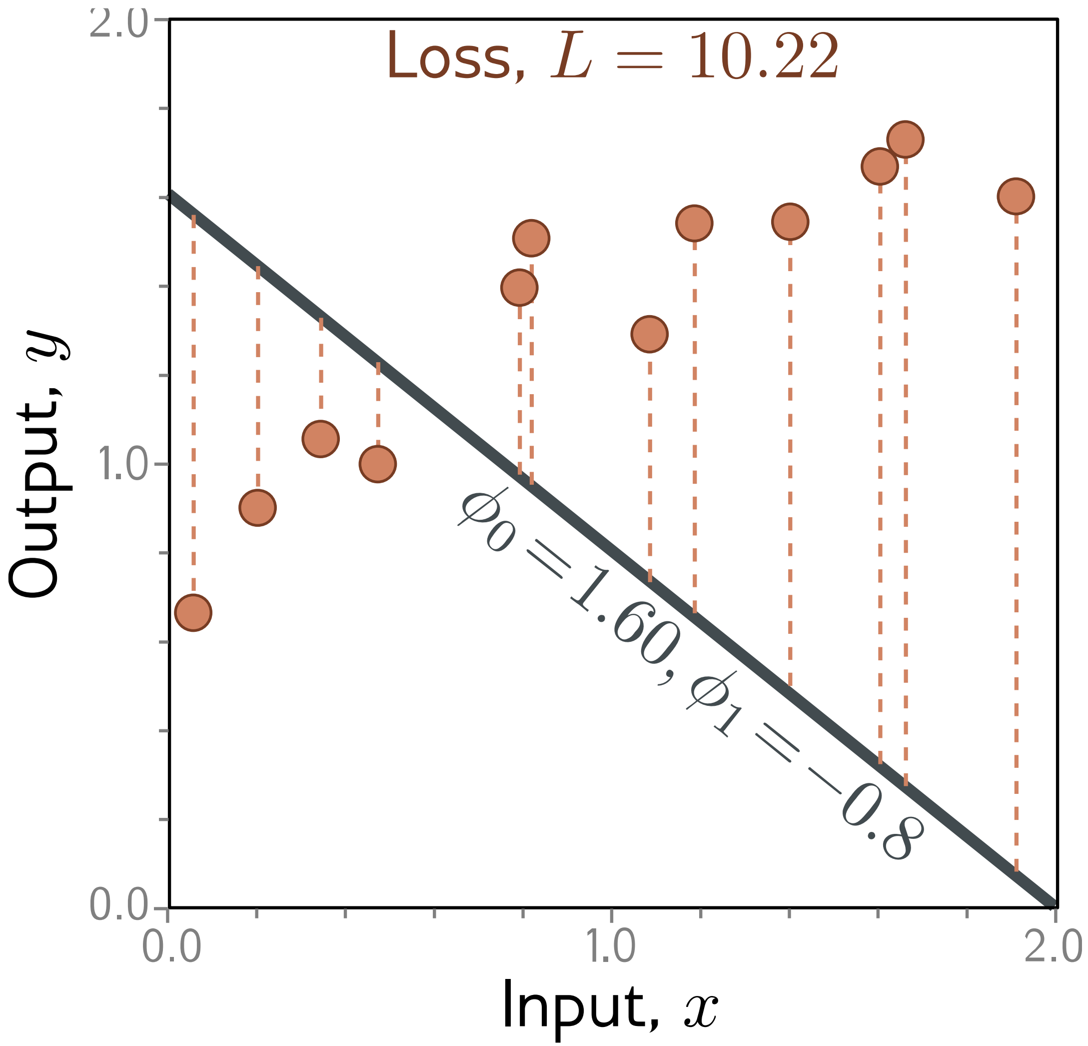
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



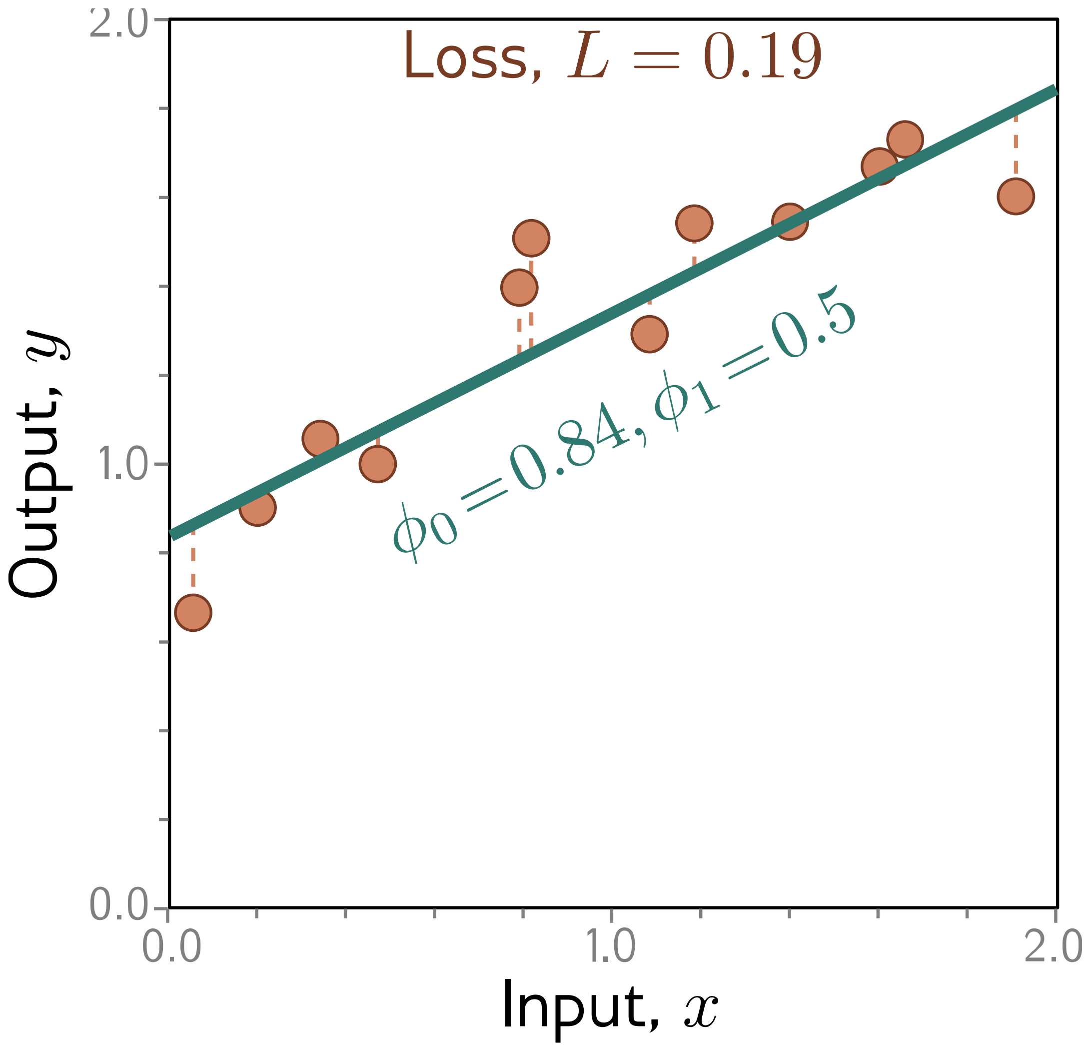
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



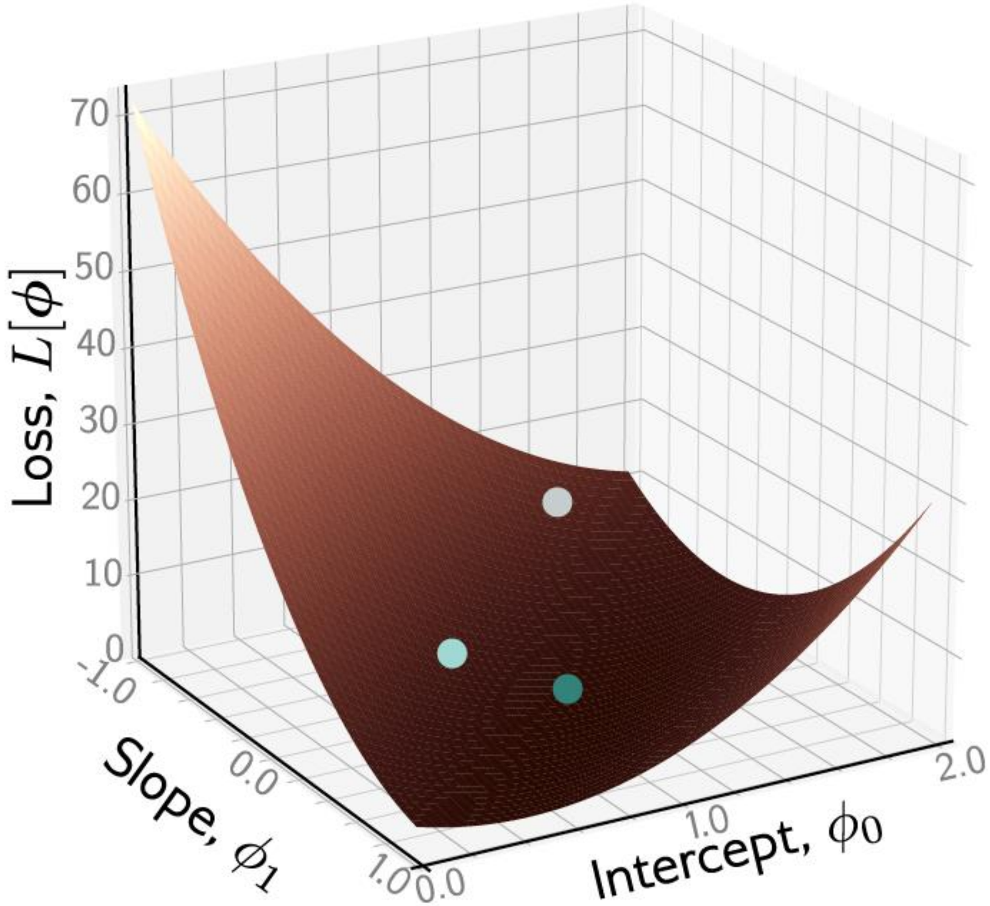
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function

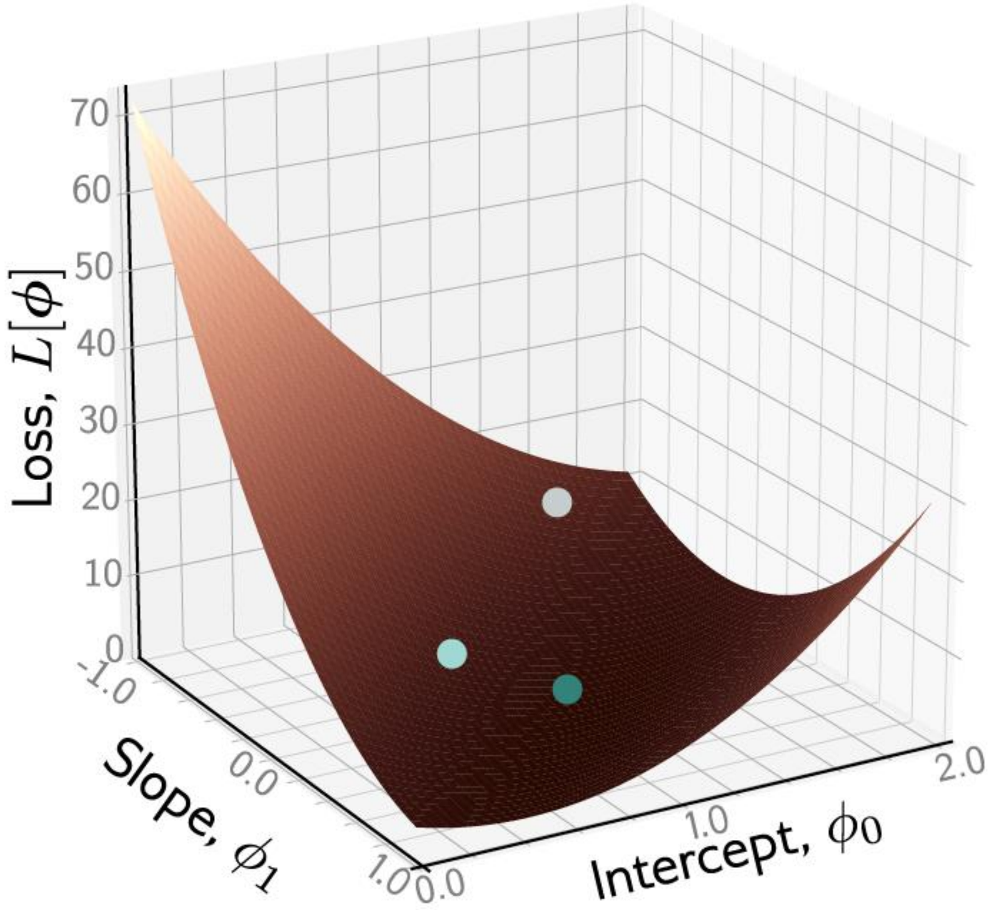


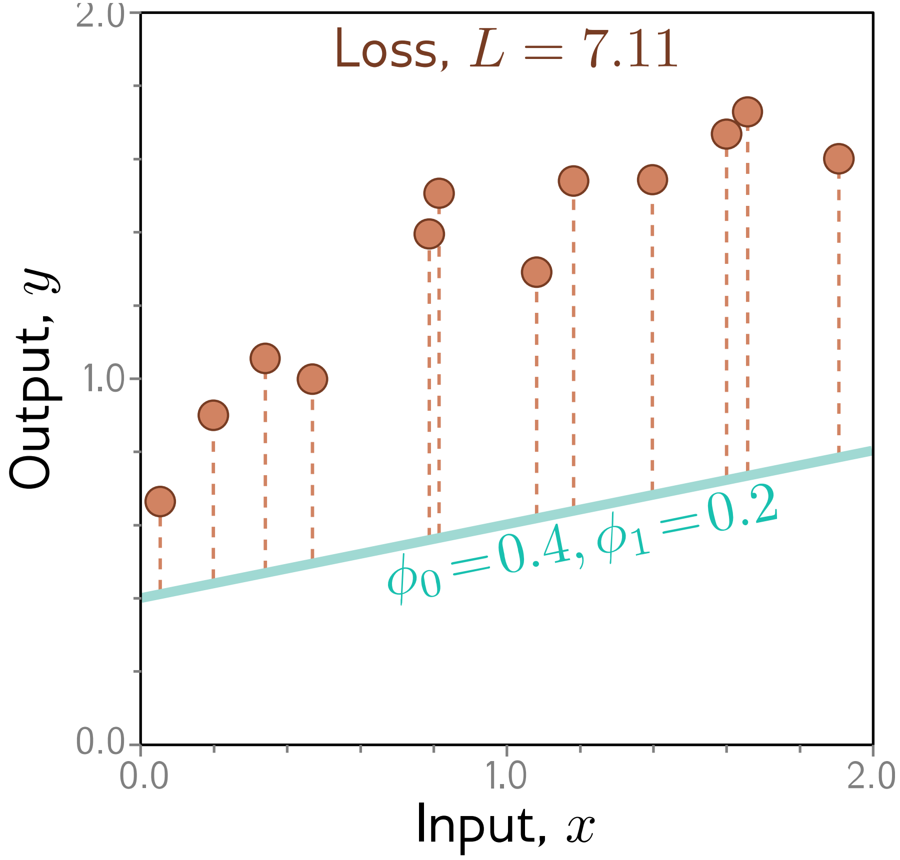
Loss function:

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

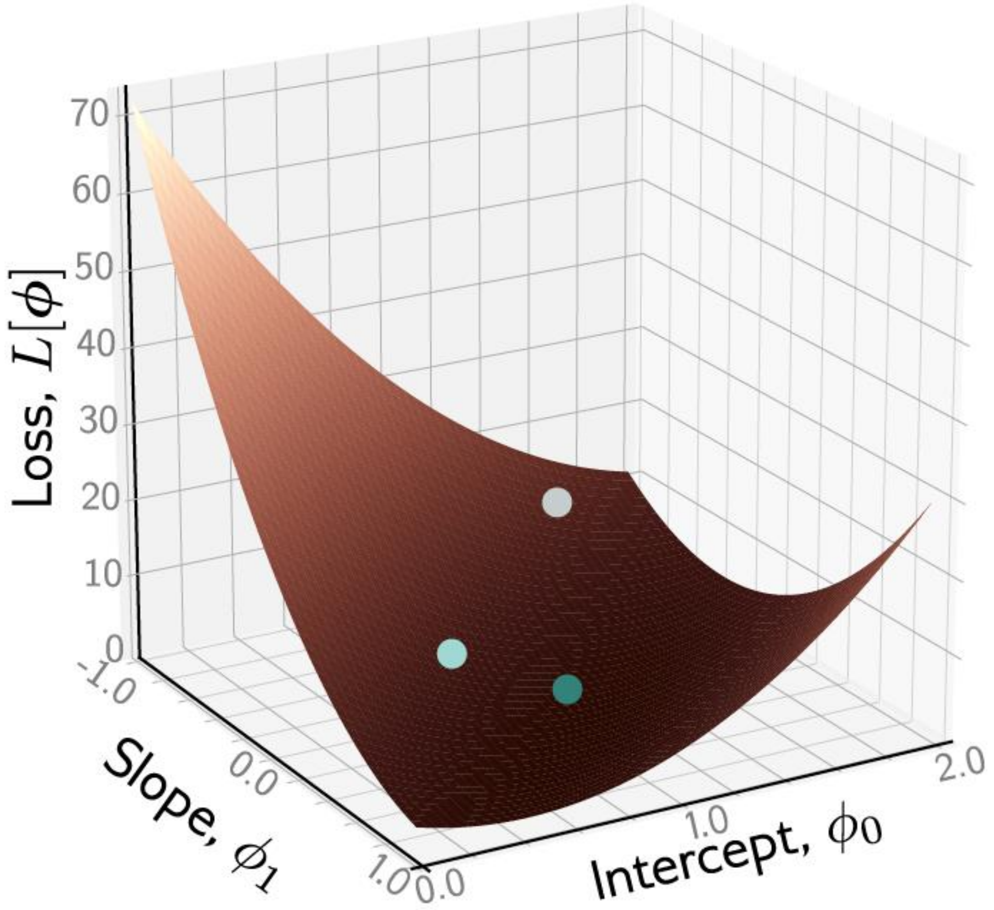
"Least squares loss function"

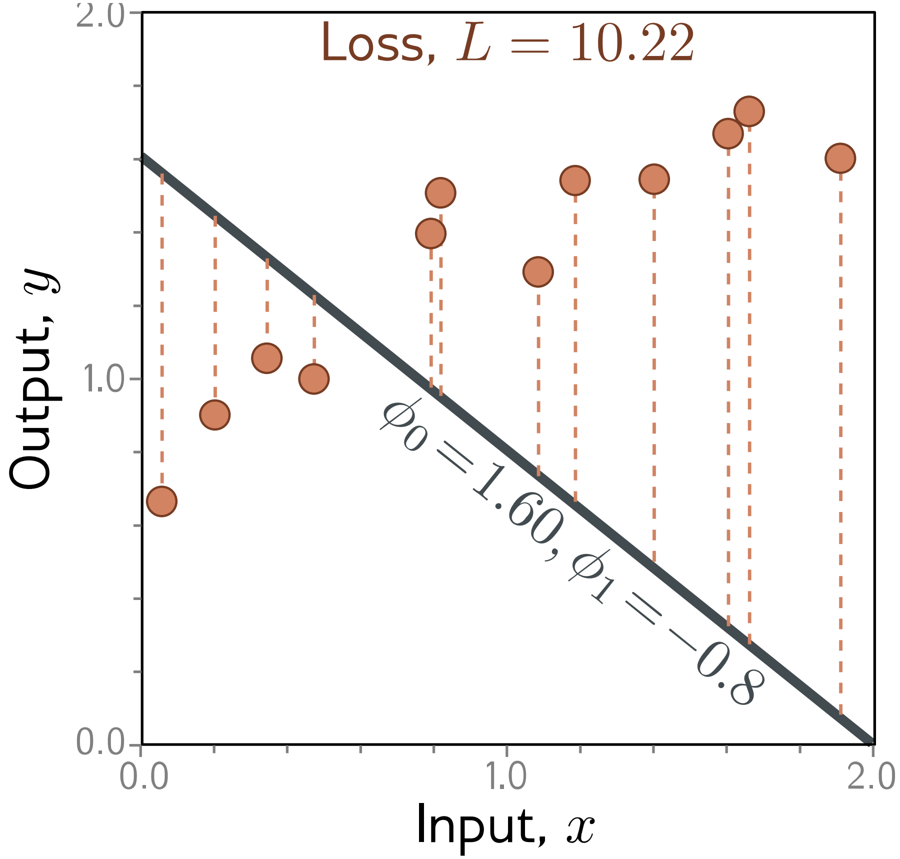
Example: 1D Linear regression loss function



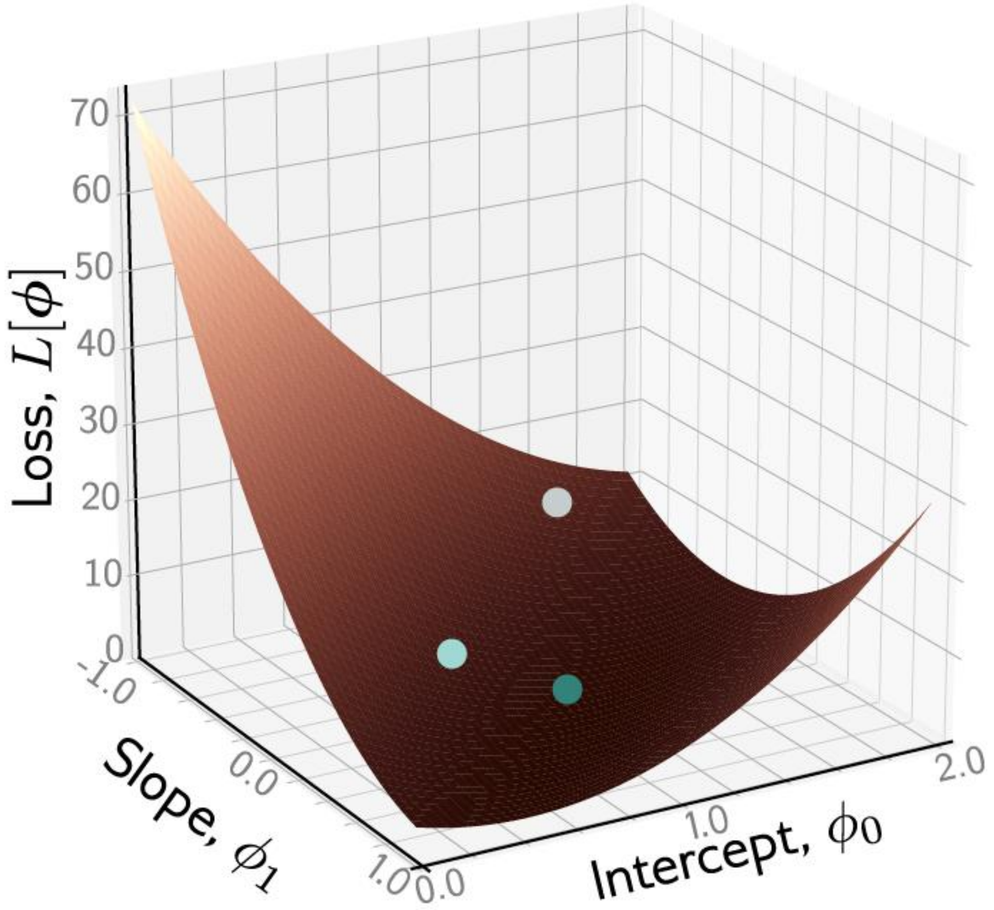


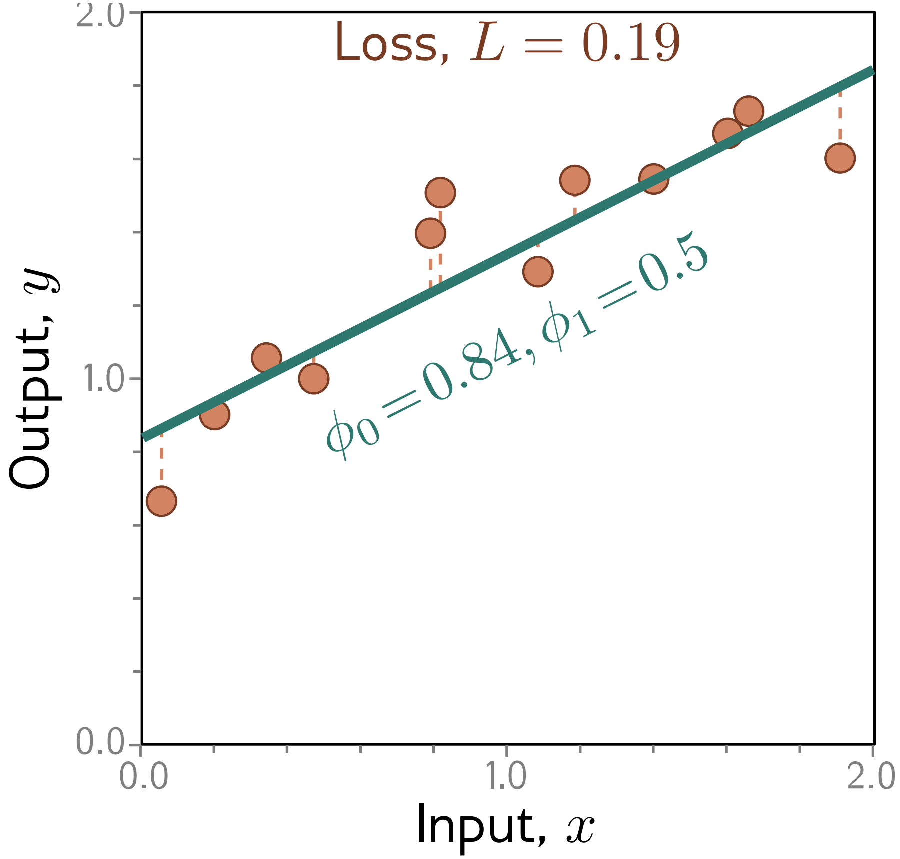
Example: 1D Linear regression loss function



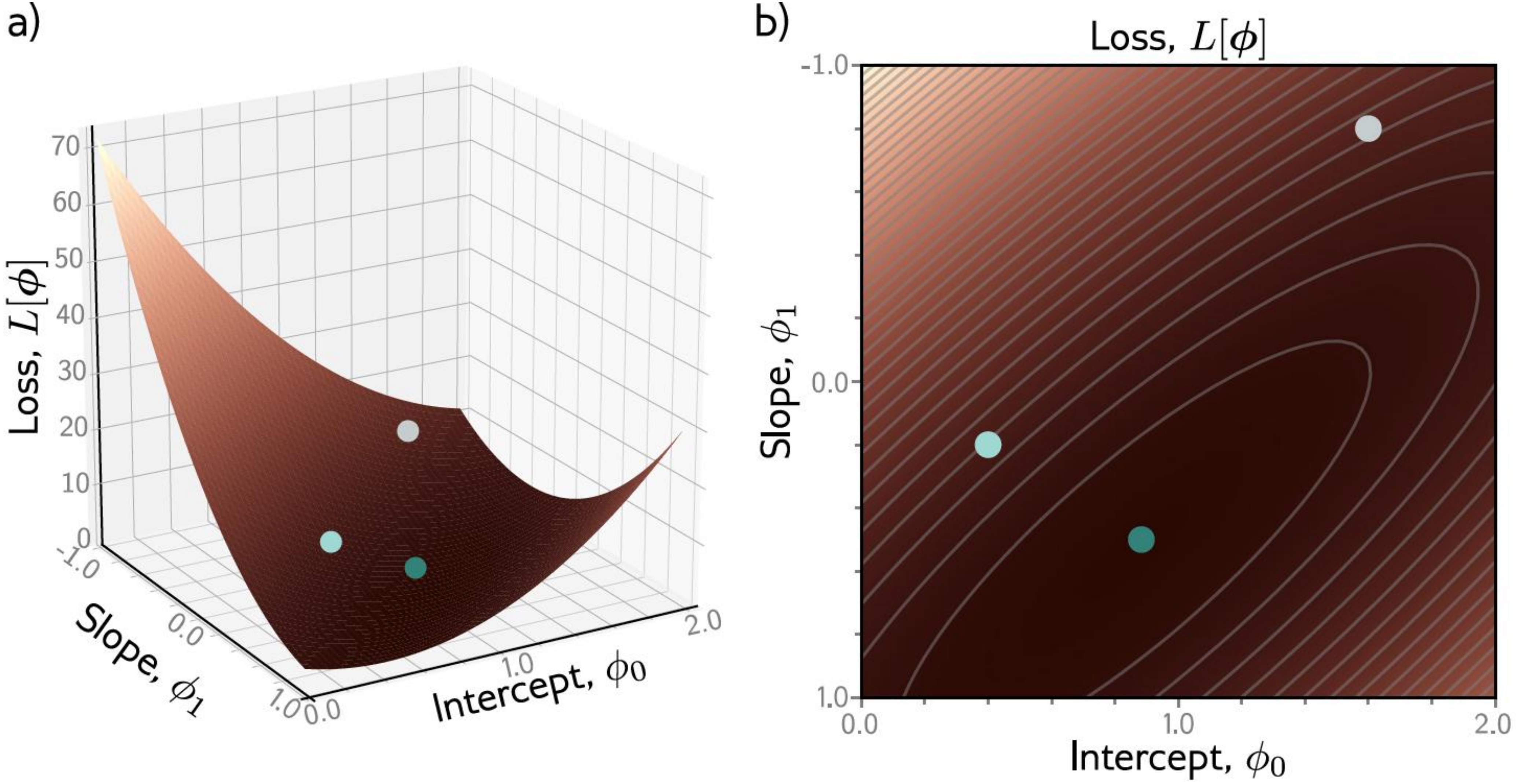


Example: 1D Linear regression loss function

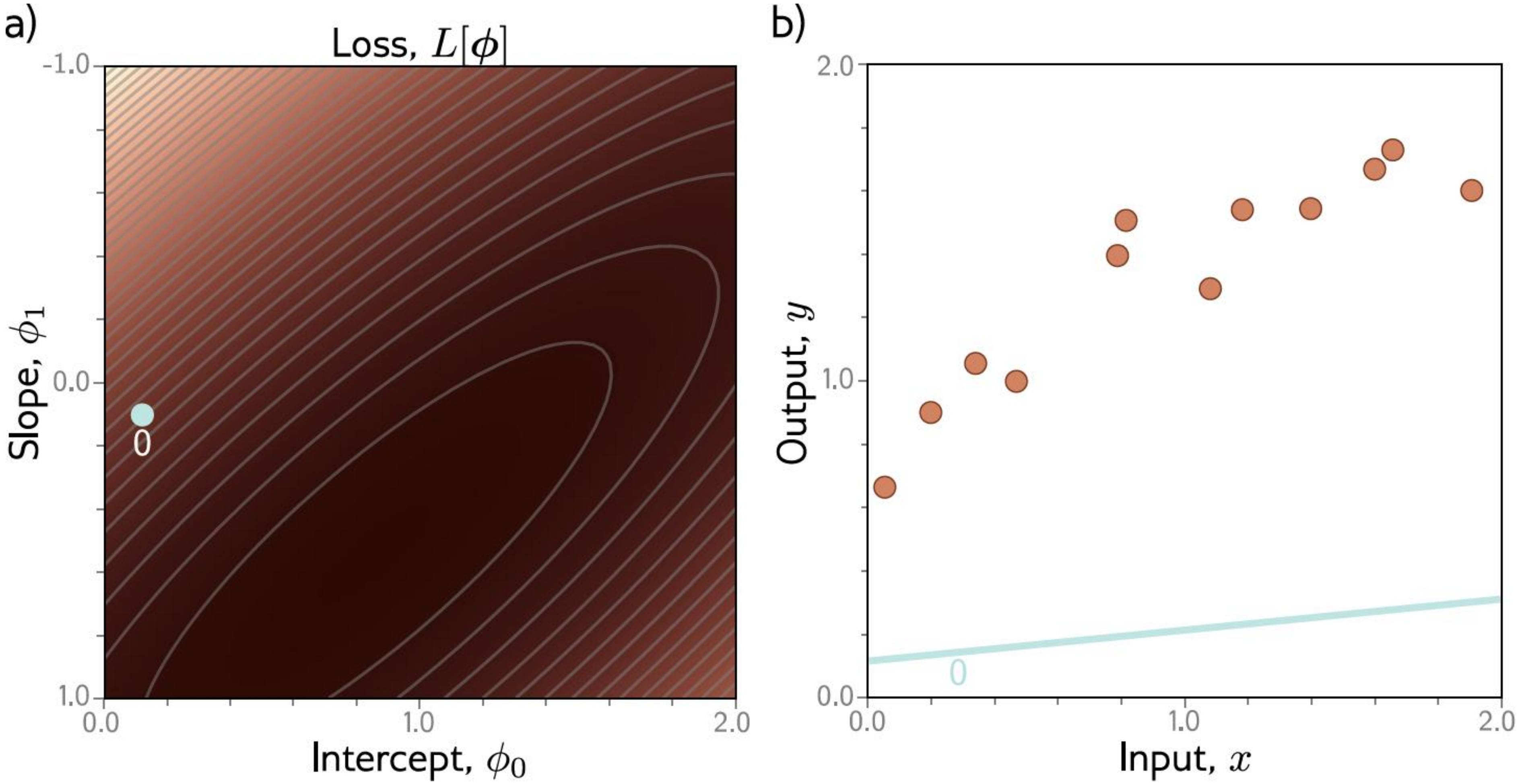




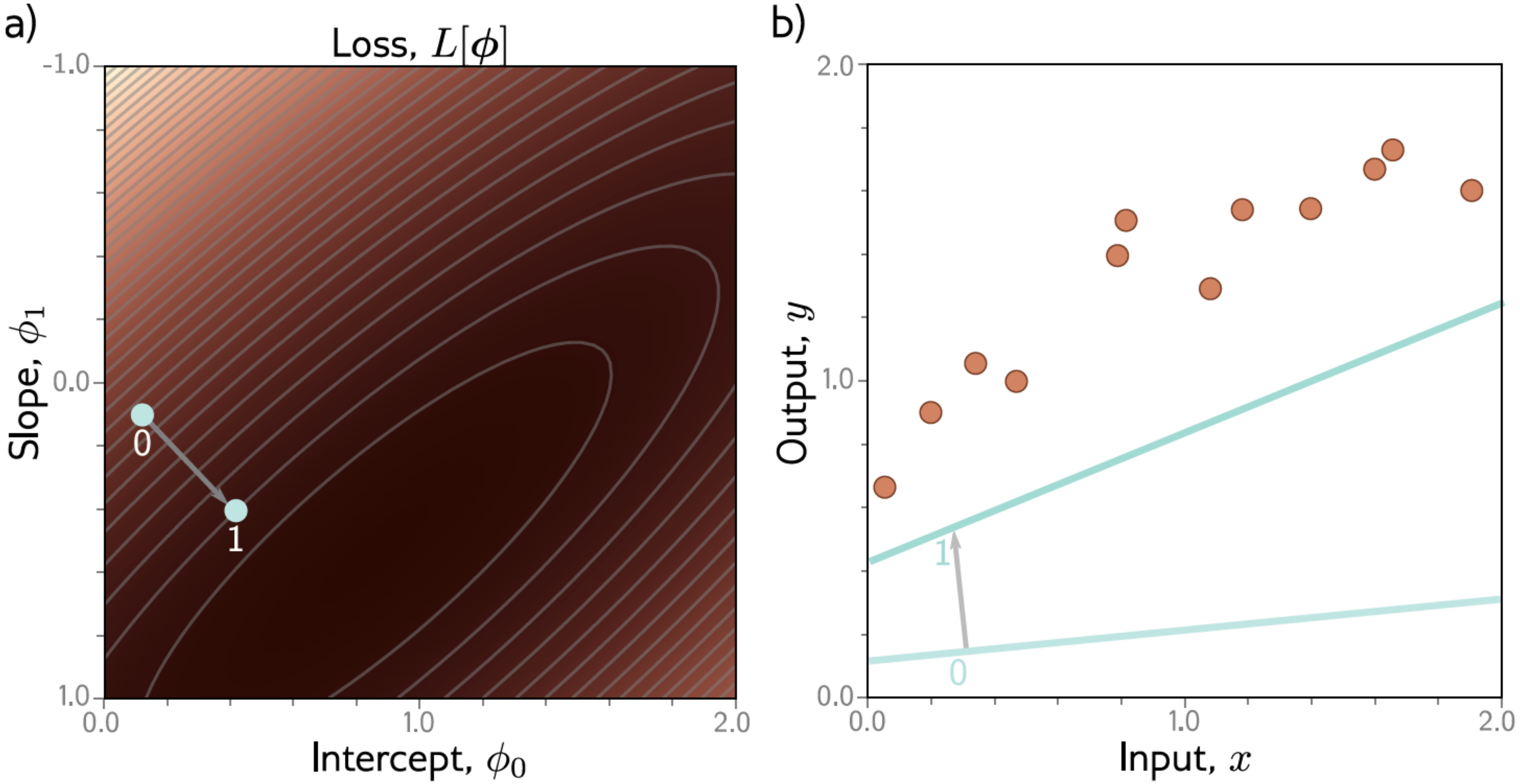
Example: 1D Linear regression loss function



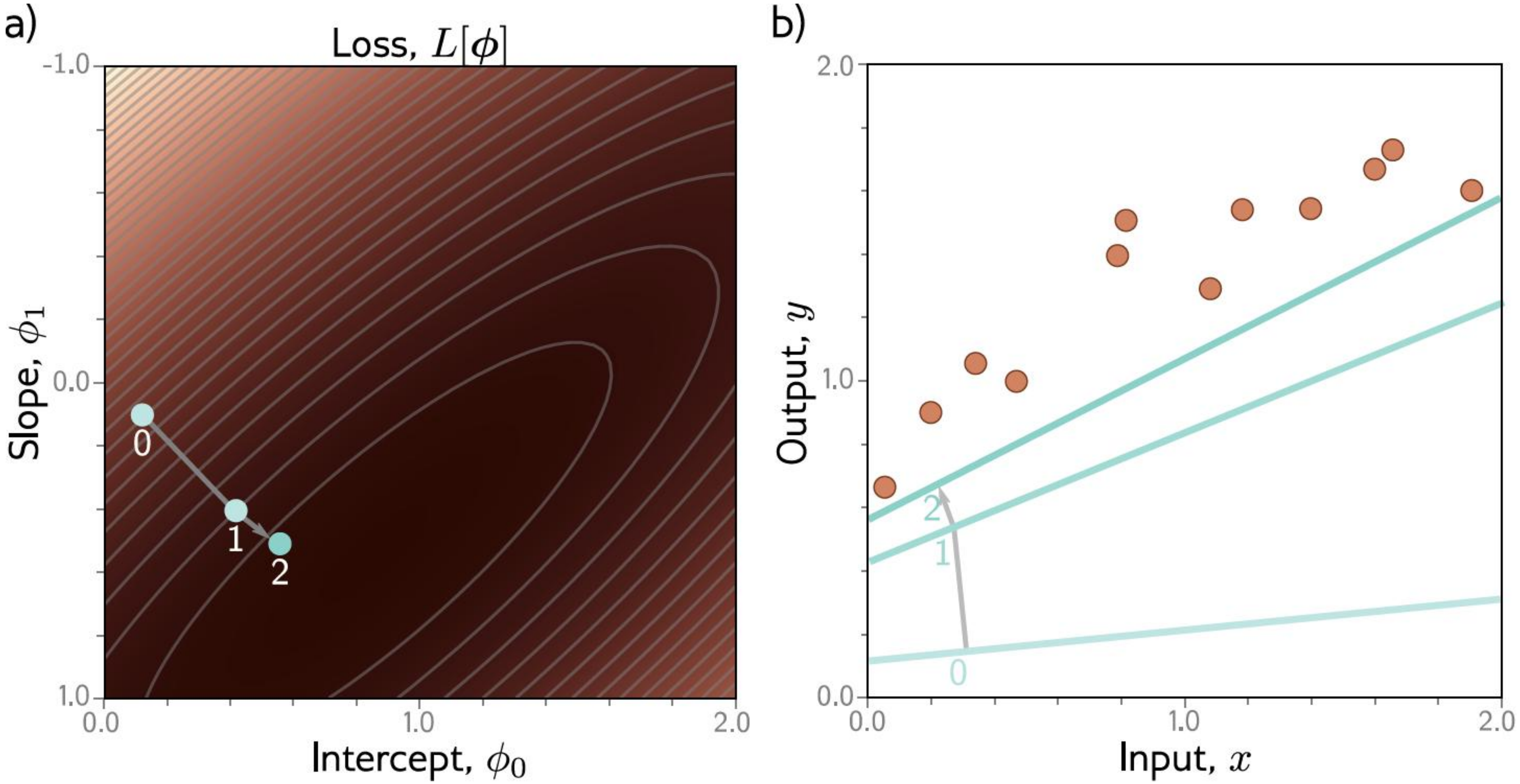
Example: 1D Linear regression training



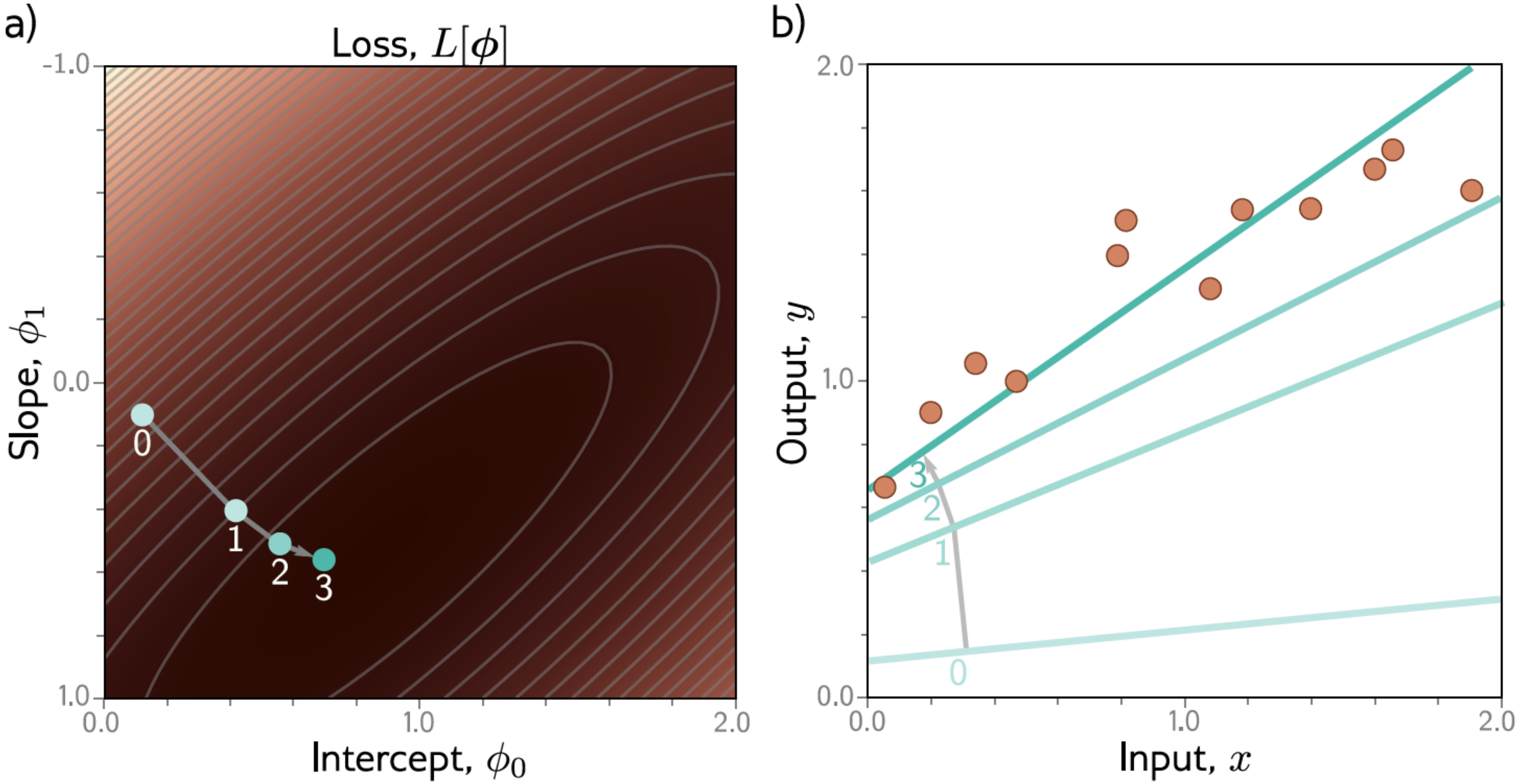
Example: 1D Linear regression training



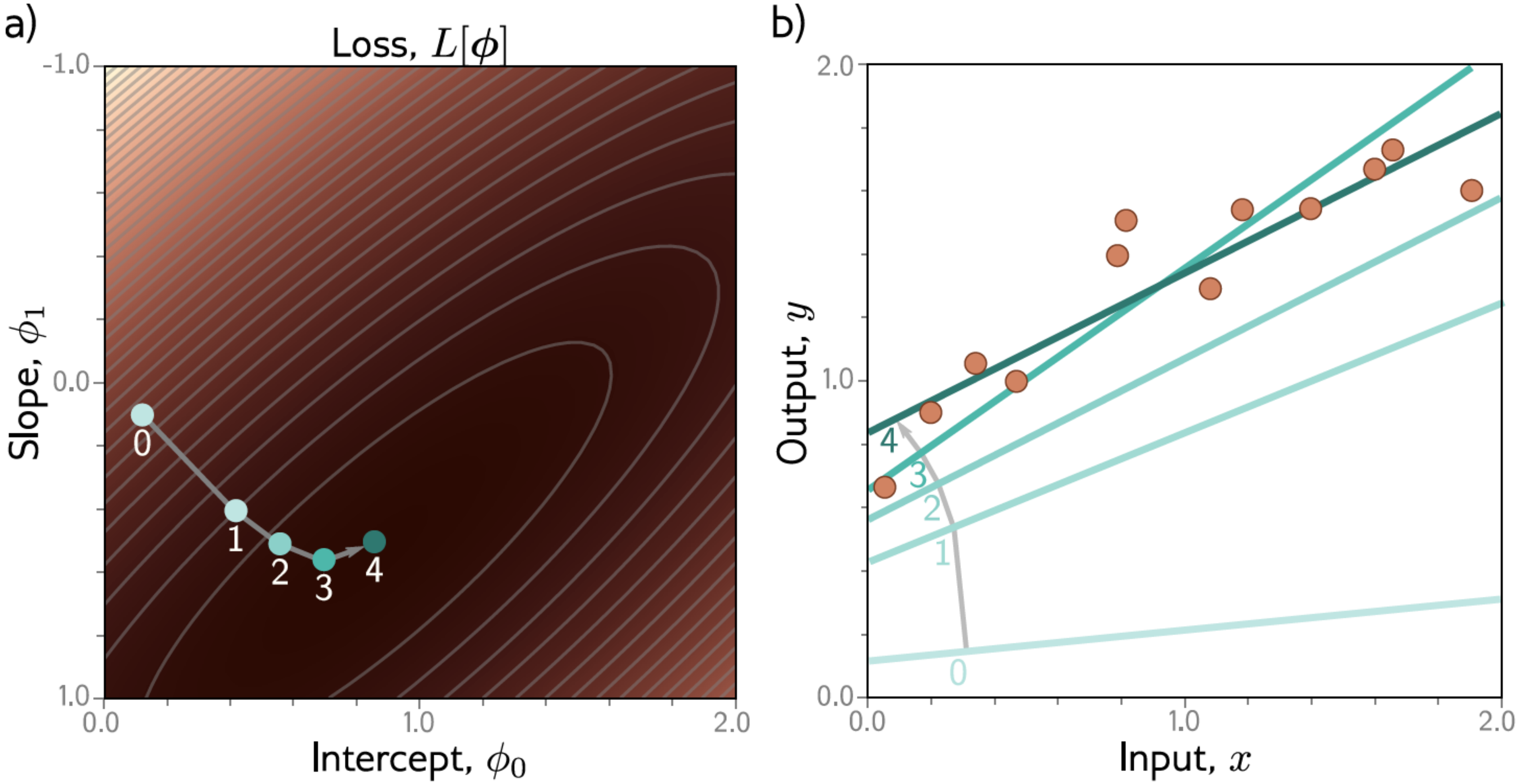
Example: 1D Linear regression training



Example: 1D Linear regression training



Example: 1D Linear regression training



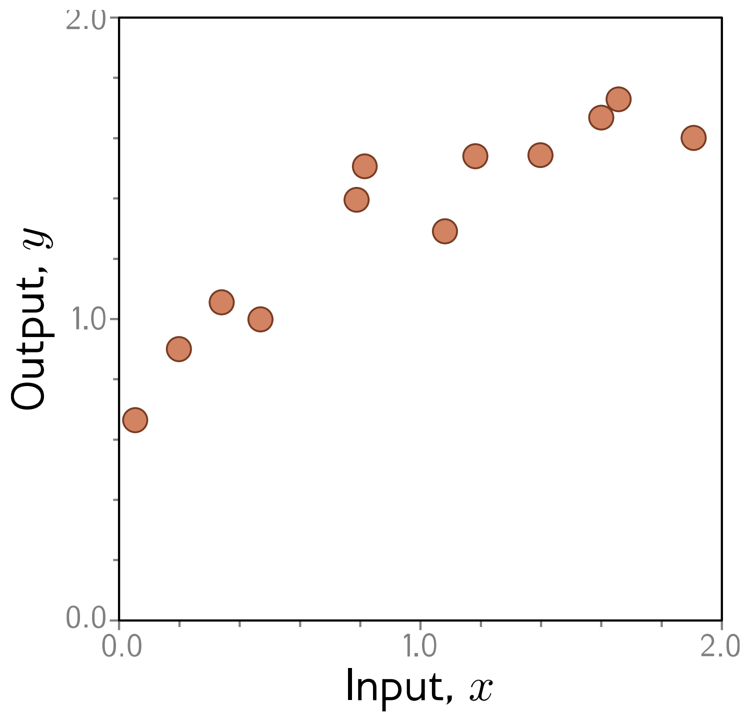
This technique is known as gradient descent

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 a million parameters
- When is a Closed-Form Solution Not Possible?
  - A closed-form solution is difficult or impossible to derive due to:
    - Non-linearities (e.g., deep learning models)
    - Constraints (e.g., regularization penalties)
    - High-dimensional optimization (e.g., logistic regression, SVM)
  - In such cases, iterative methods (e.g., gradient descent, Newton’s method) are used to approximate the solution.

Example: 1D Linear regression testing

- Test with different set of paired 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?

- 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)