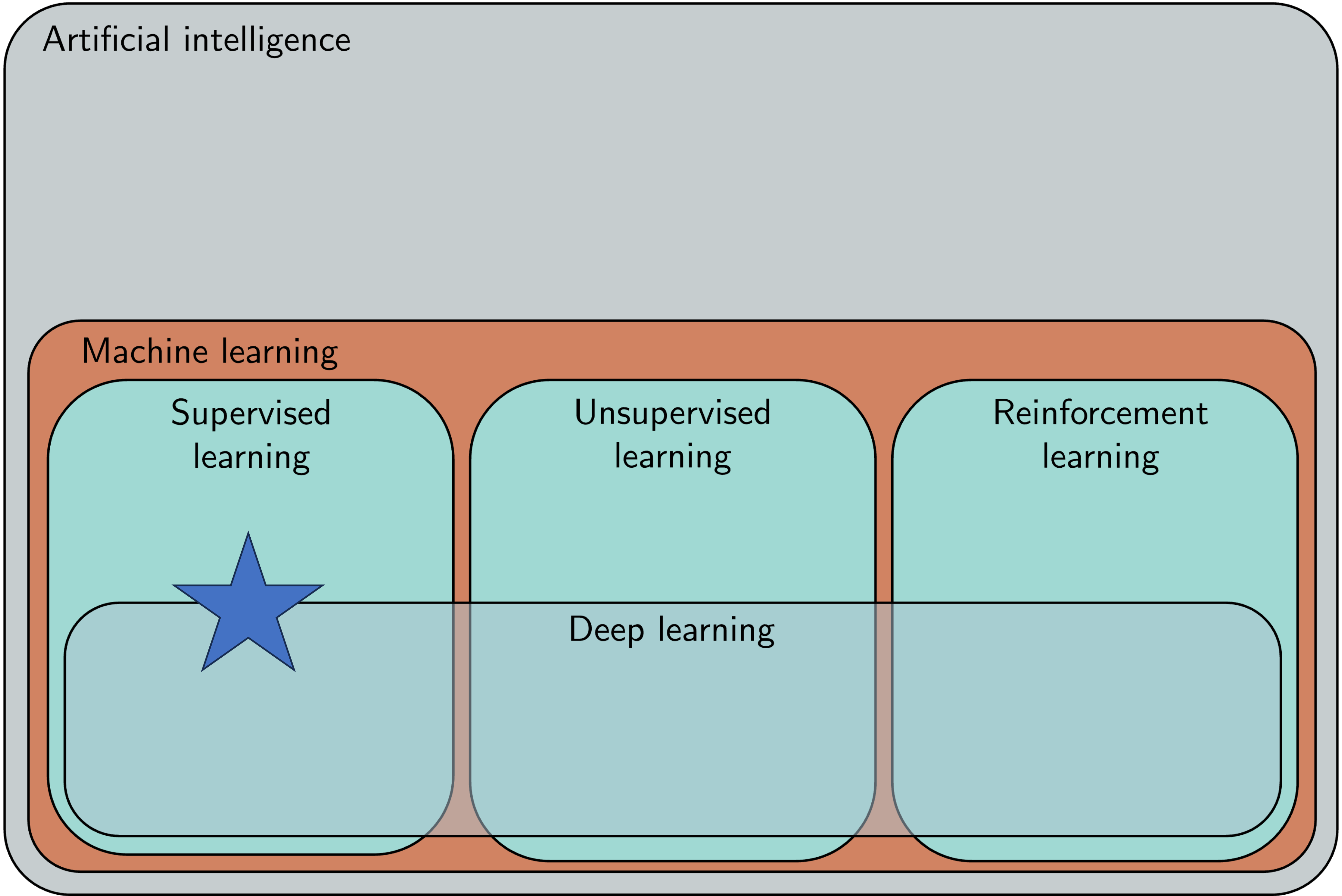
BOSTON UNIVERSITY

Supervised Learning Terminology and Concepts

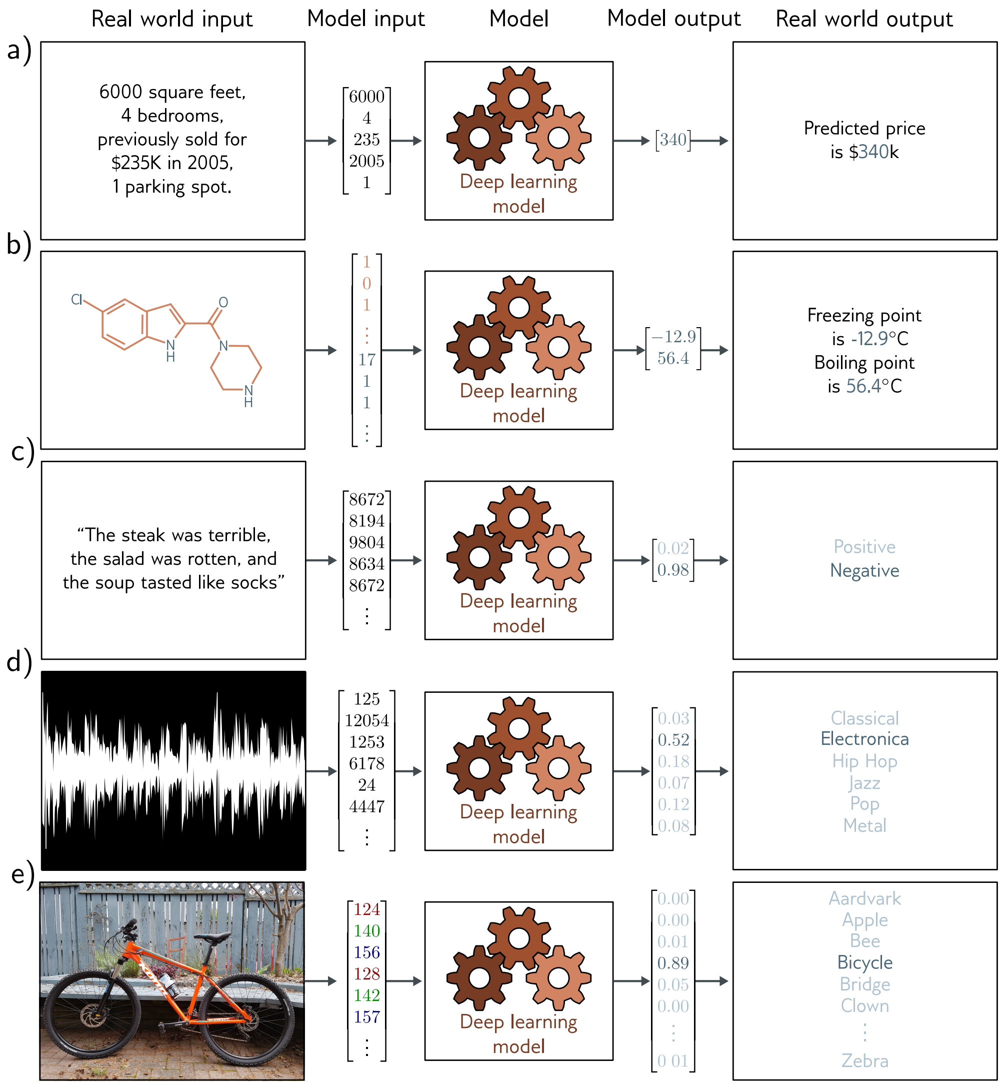
DL4DS Spring 2024

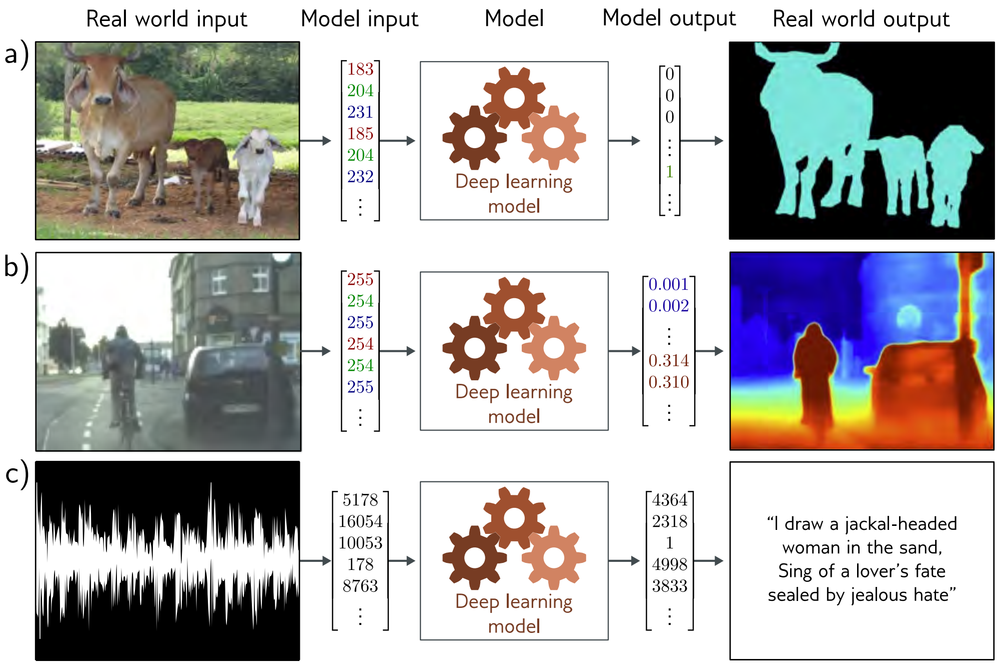
Lecture Outline

- Homeworks and Jupyter Notebooks plan
- Supervised Learning
- More on Projects

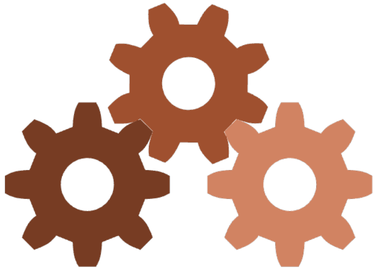

Artificial intelligence
Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Deep learning

Supervised Learning Classification and Regression Applications





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?

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 family of equations → “inductive bias”
- 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 training and evaluation datasets 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?

Also Appendix A of the book.

Notation:

- Input:

$$\mathbf{x}$$

- Output:

$$\mathbf{y}$$

- Model:

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

Variables always Roman letters

Normal lower case = scalar
Bold lower case = vector
Capital Bold = matrix

Functions always square brackets

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

Notation example:

- Input:

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

Vector: Structured or tabular data

- Output:

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

Scalar output

- Model:

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

Scalar output function (with vector input)

Model

- Parameters:

$$\phi$$

Parameters always Greek letters

- Model :

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

Data Set and Loss function

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

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

Data Set and 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:

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

Dataset and 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:

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

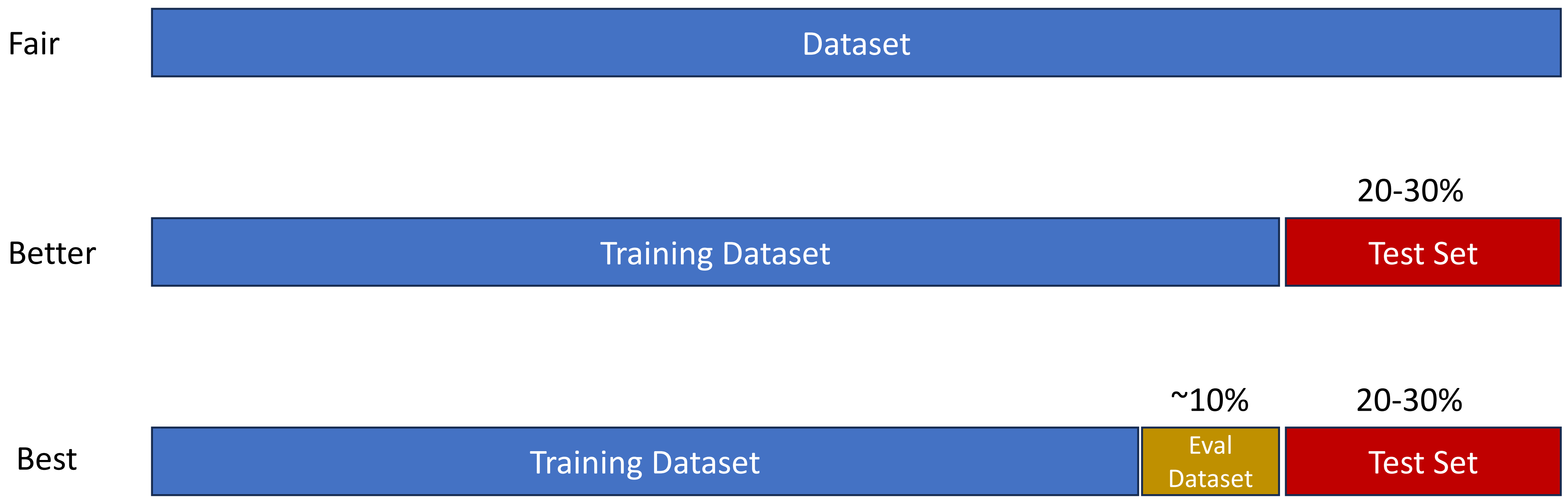
Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

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

Testing (and evaluating)

- 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:

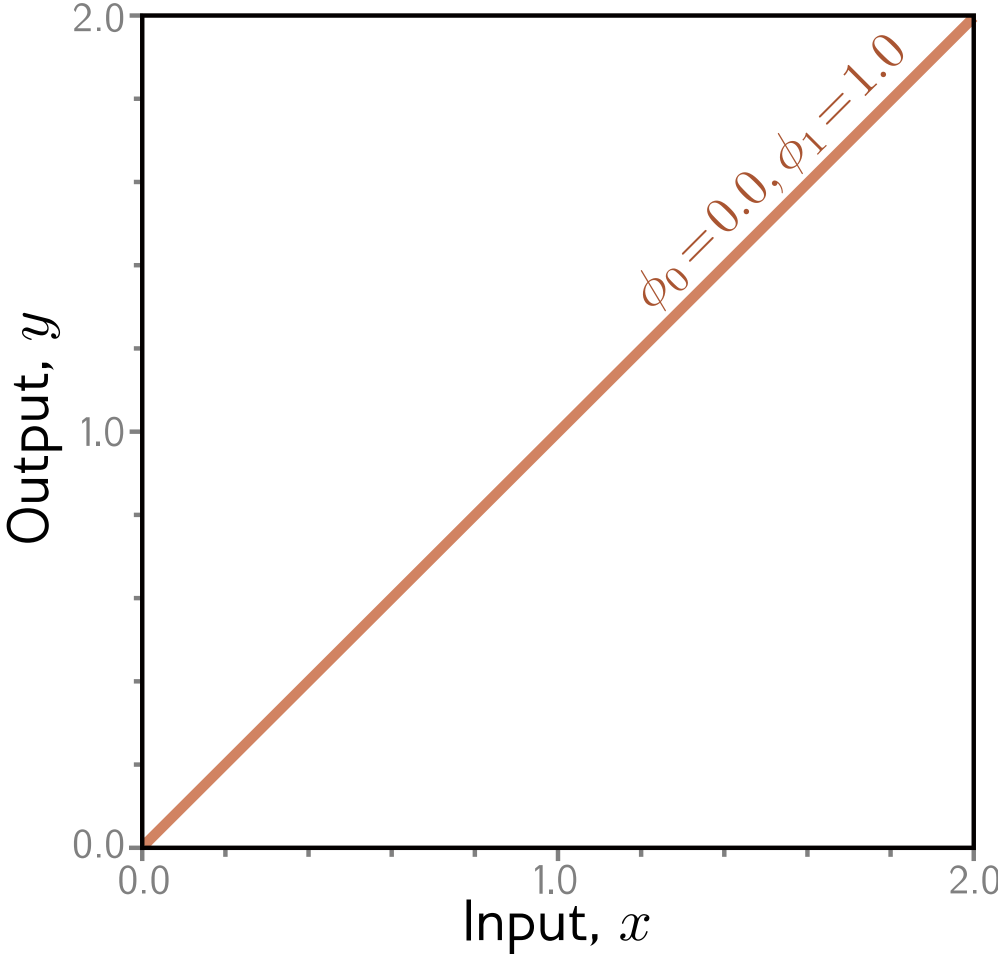
$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 x
\end{aligned}
$$

- Parameters

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

← y-offset

← slope



Example: 1D Linear regression model

- Model:

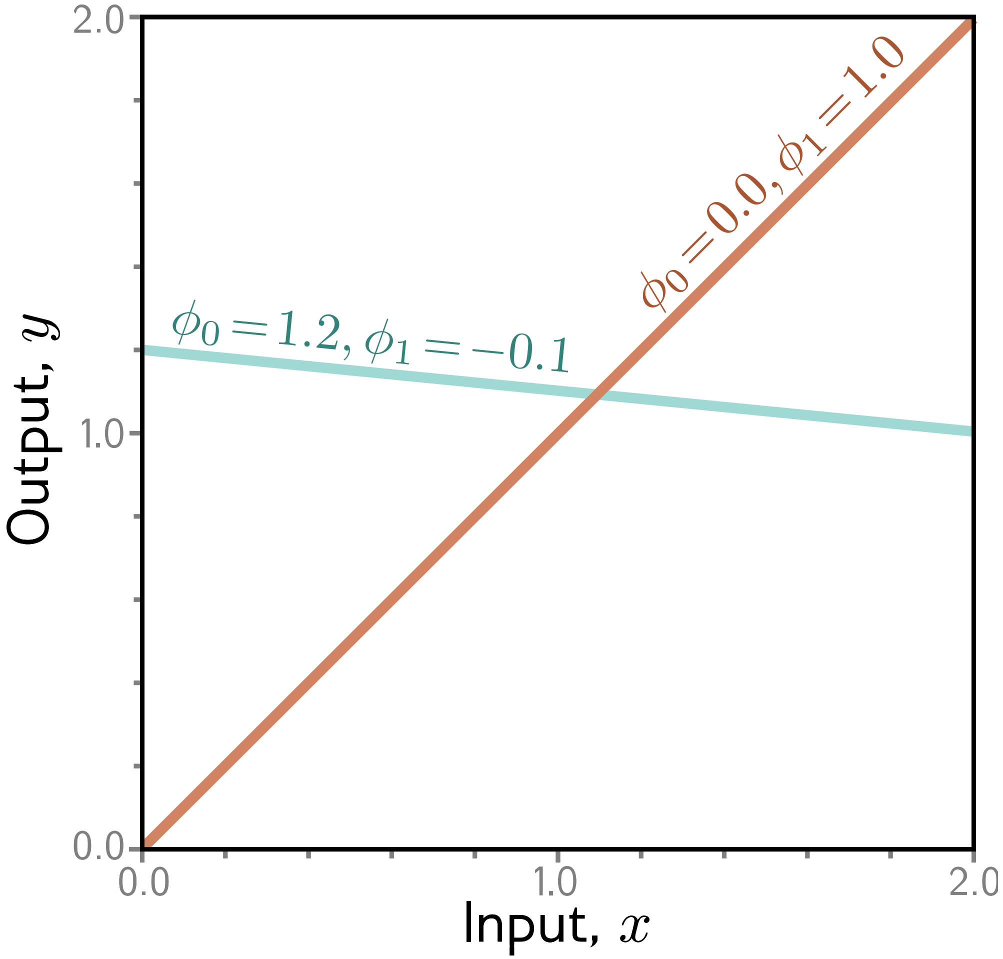
$$
\begin{aligned}
y &= \mathrm{f}[x, \boldsymbol{\phi}] \\
&= \phi_0 + \phi_1 x
\end{aligned}
$$

- Parameters

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

← y-offset

← slope



Example: 1D Linear regression model

- Model:

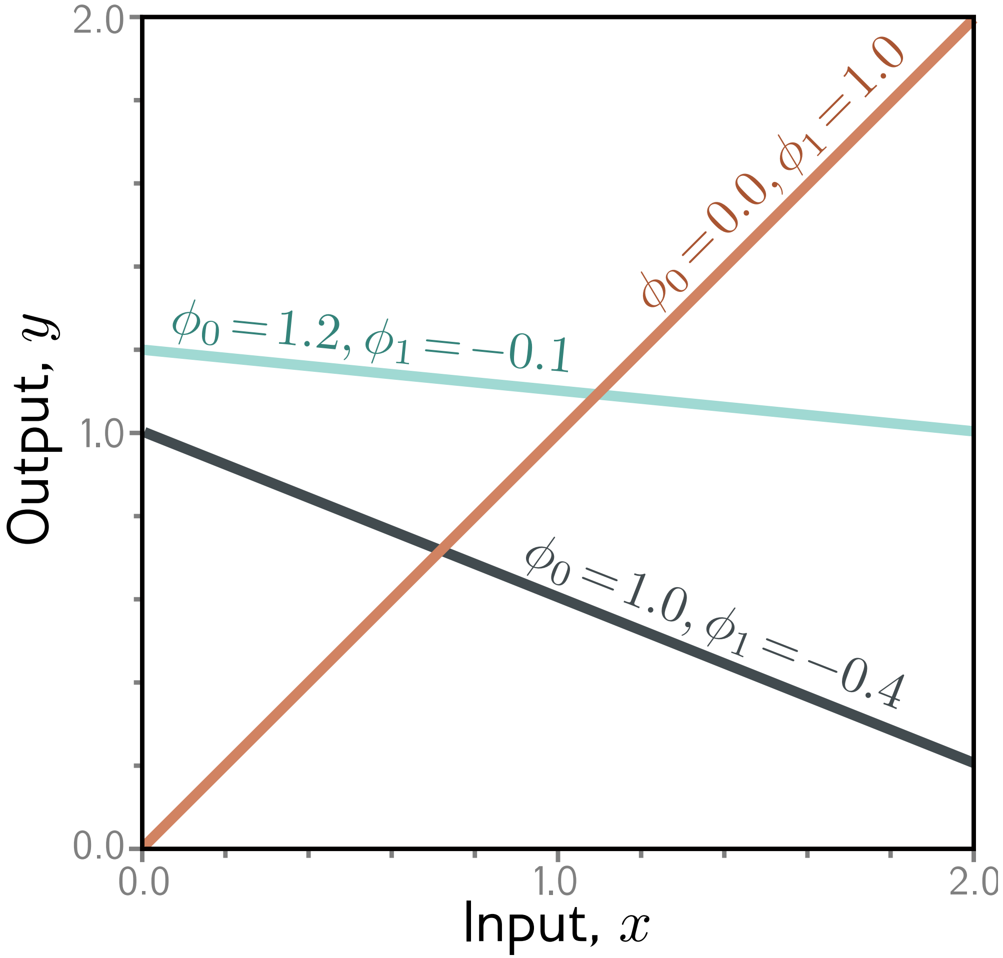
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

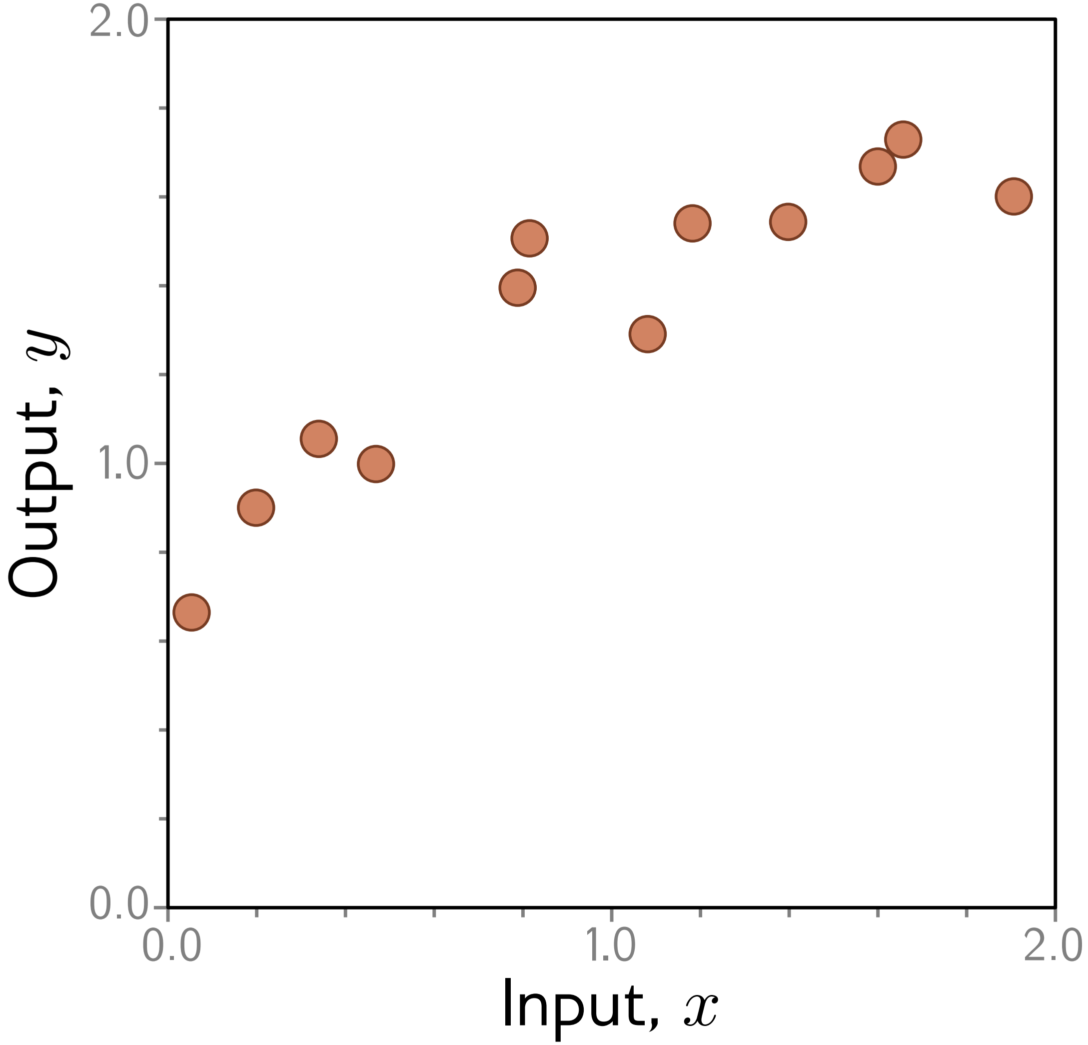
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

y-offset

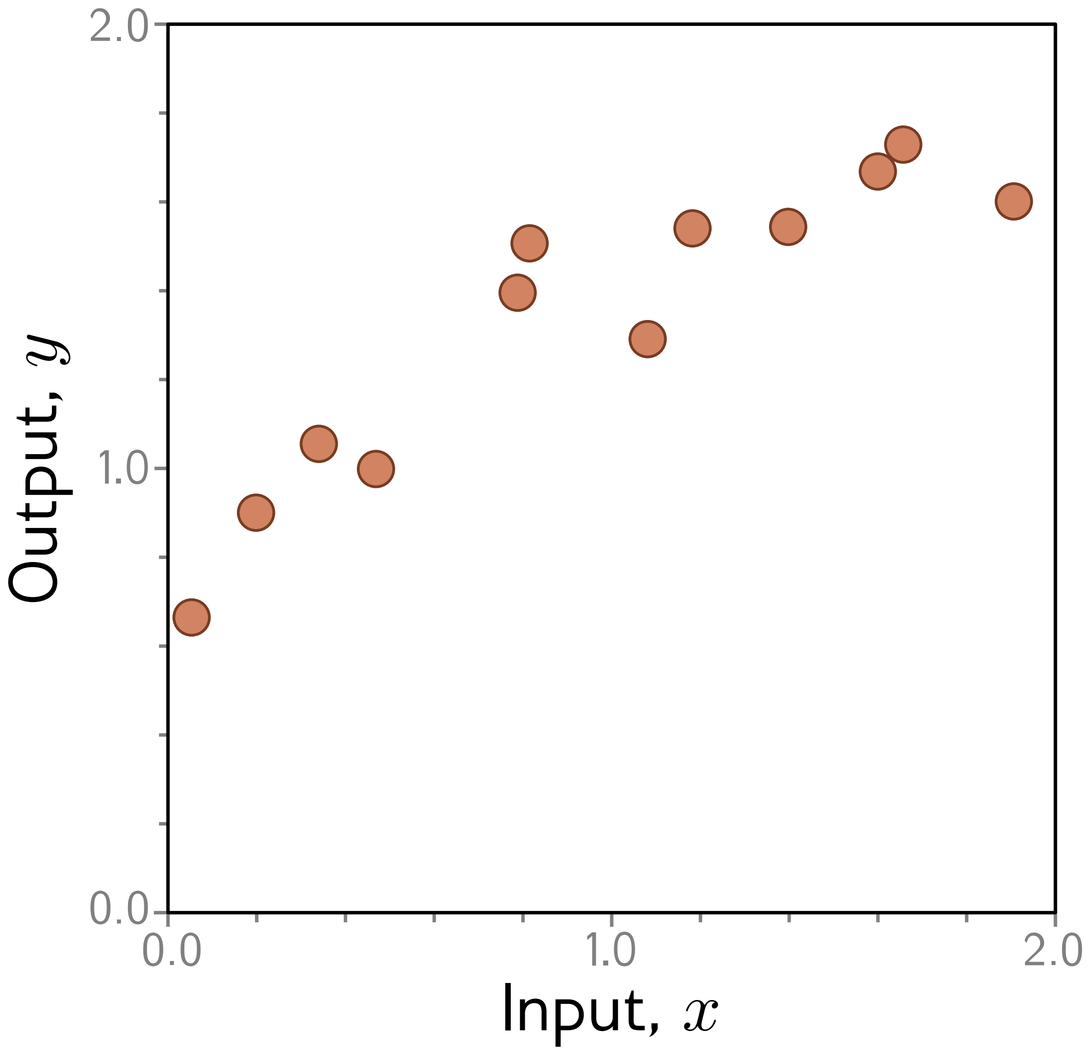
slope



Example: 1D Linear regression training data



Example: 1D Linear regression training data

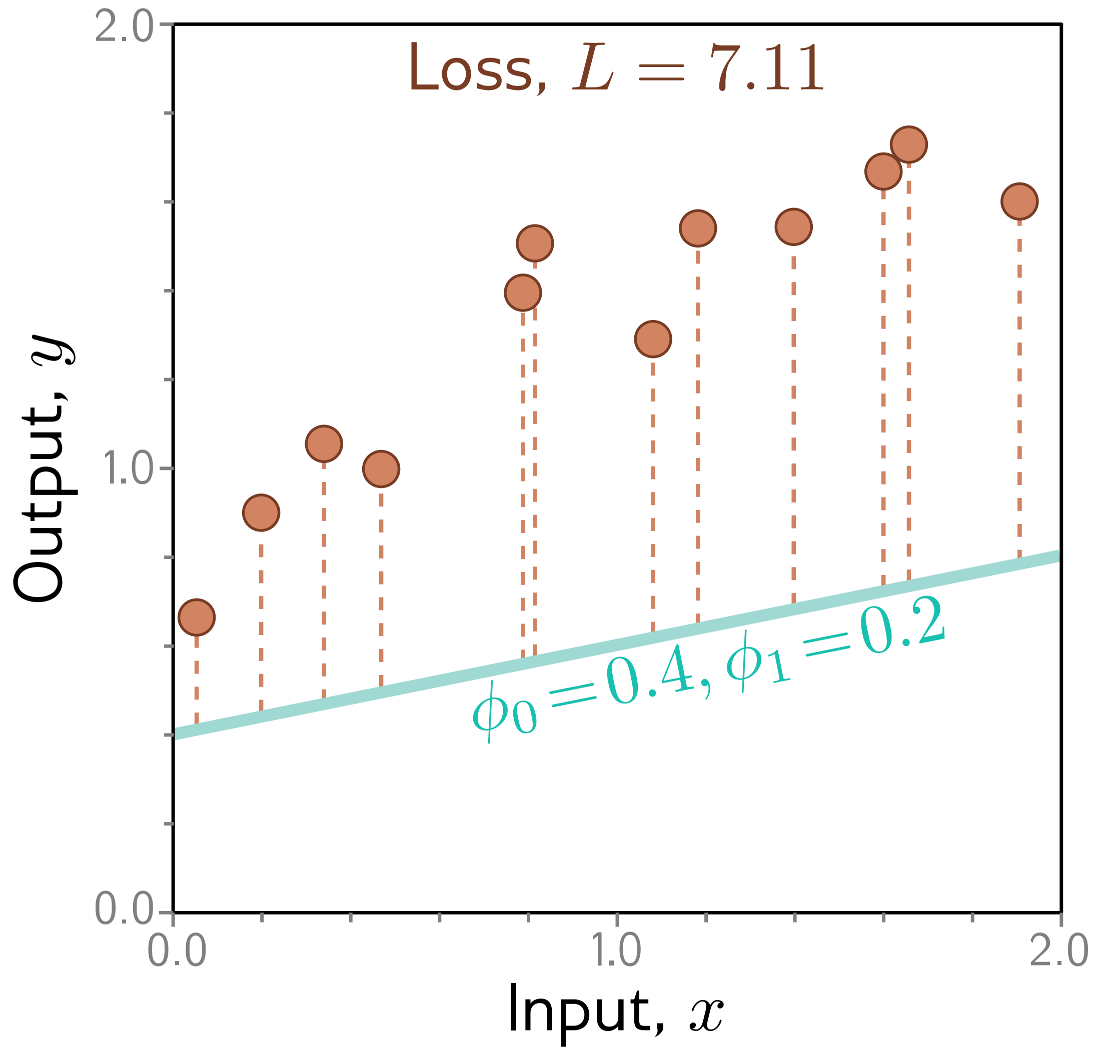


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression loss function

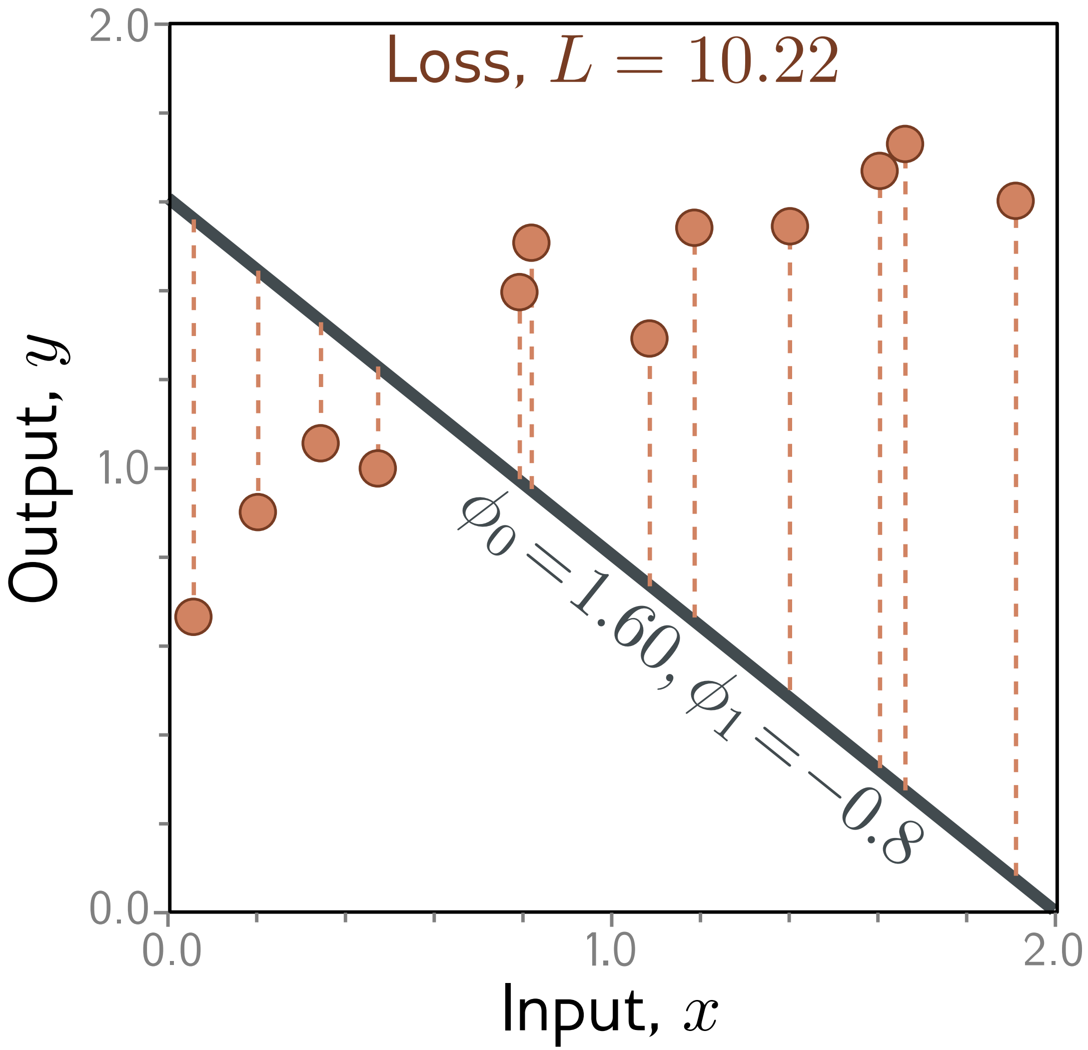


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression loss function



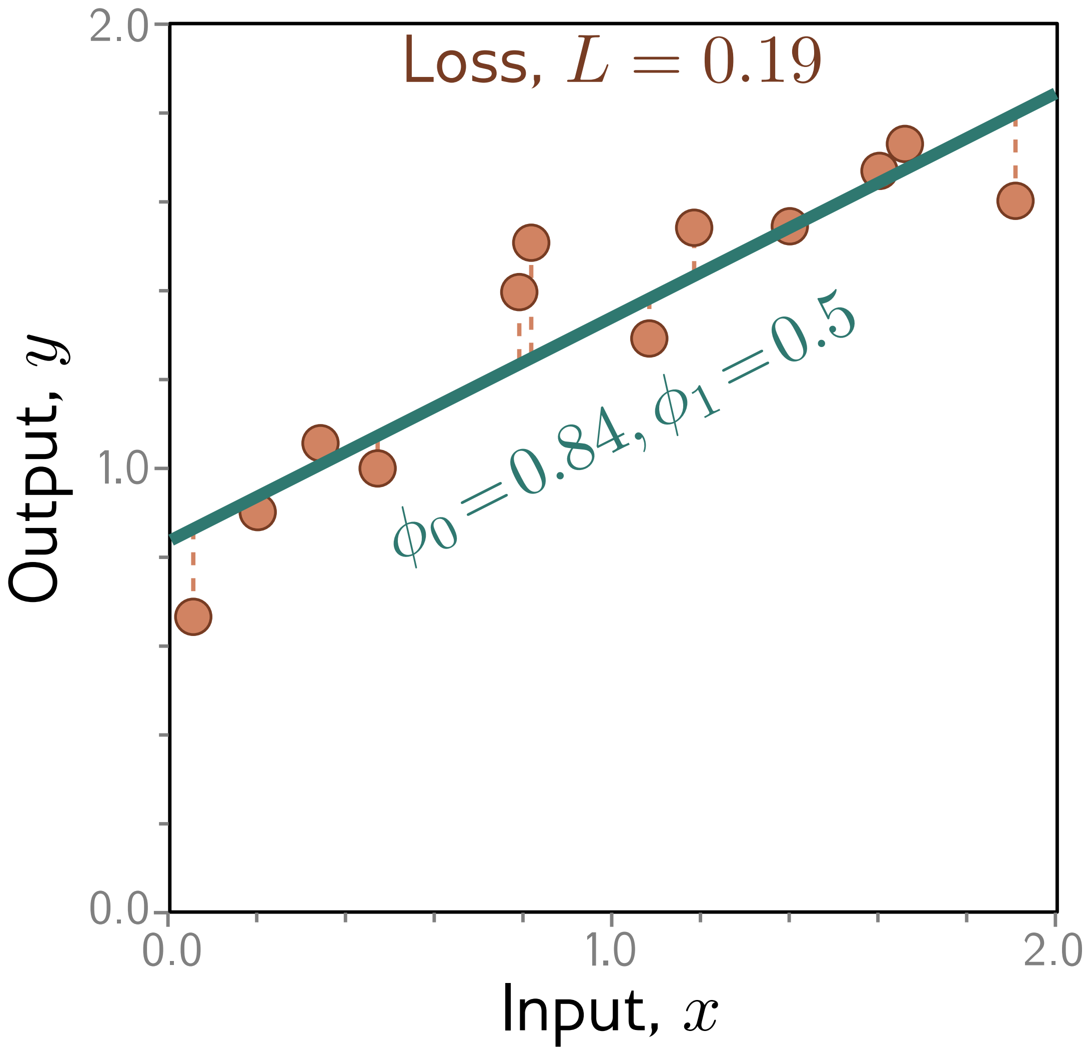
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



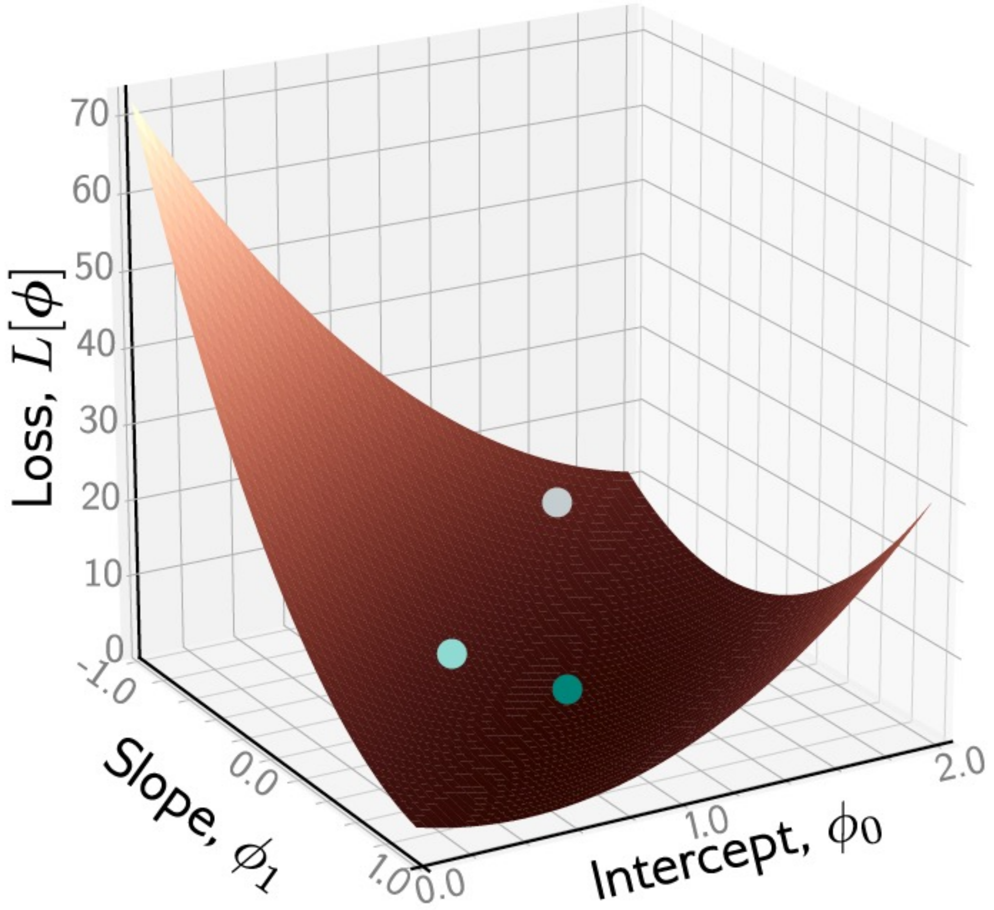
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



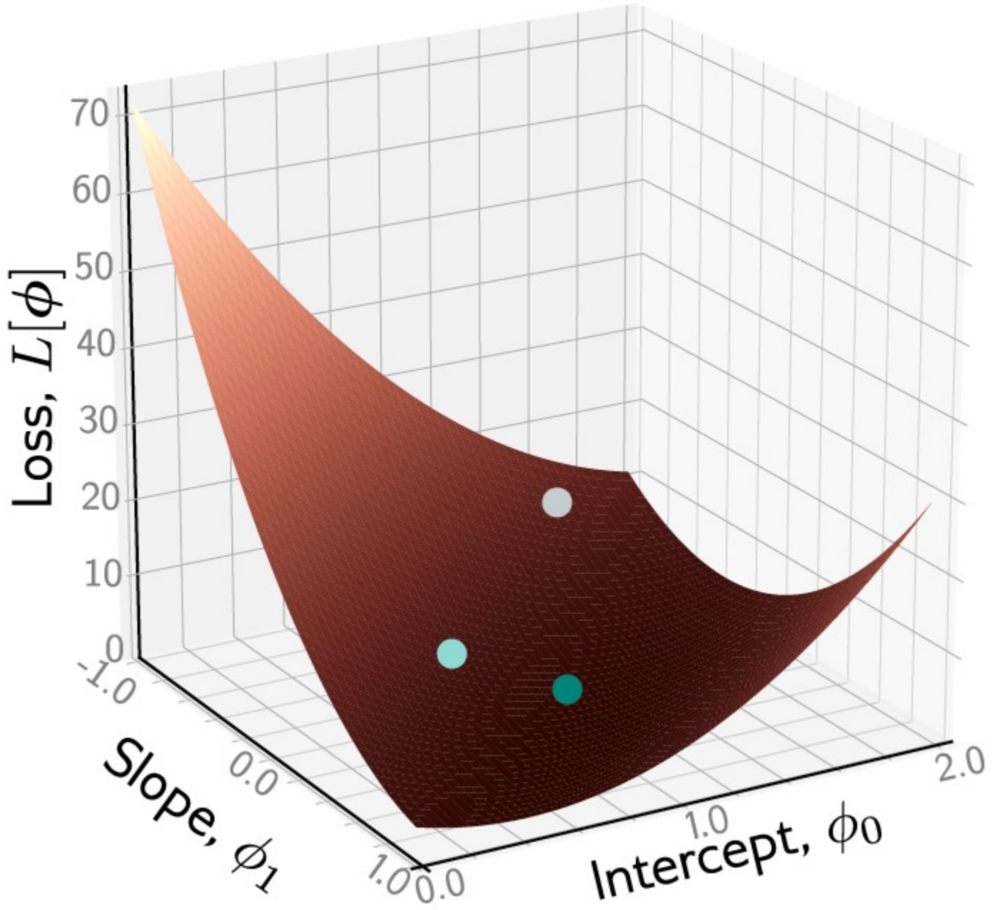
Loss function:

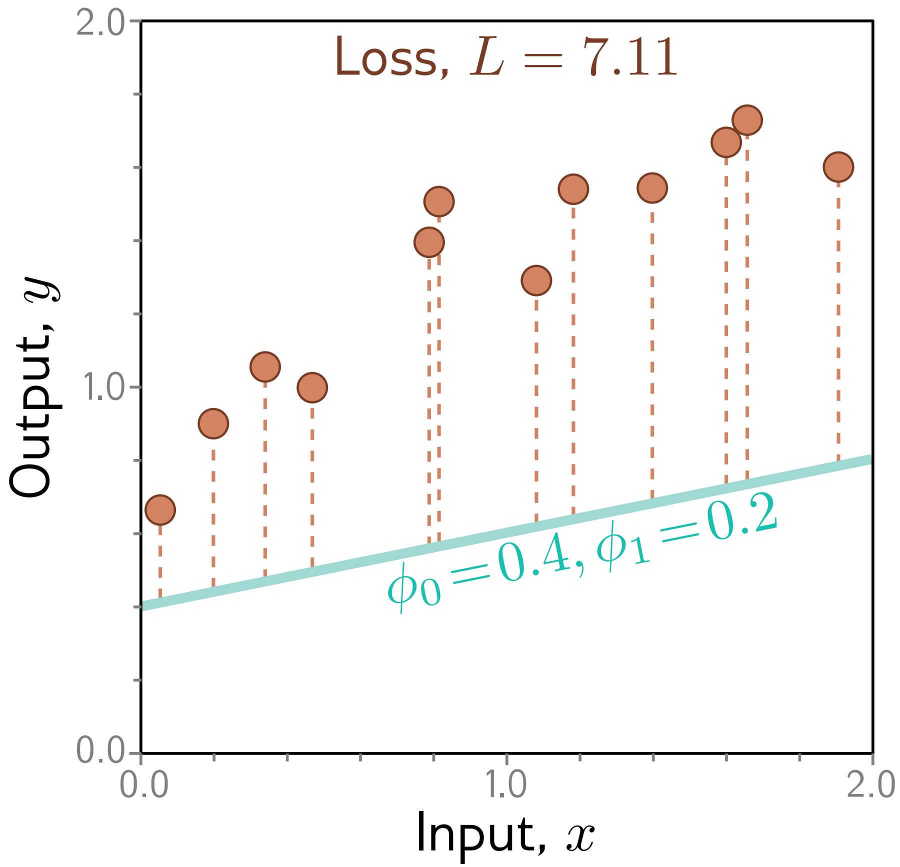
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

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

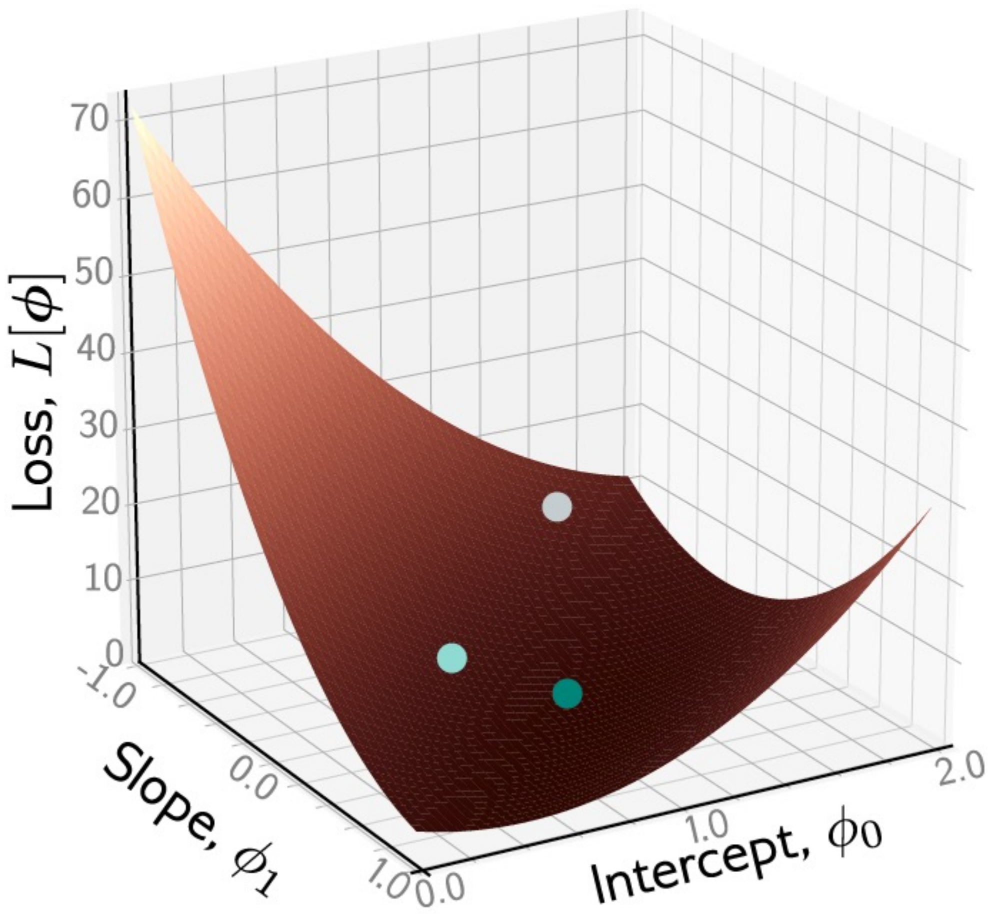
"Least squares loss function"

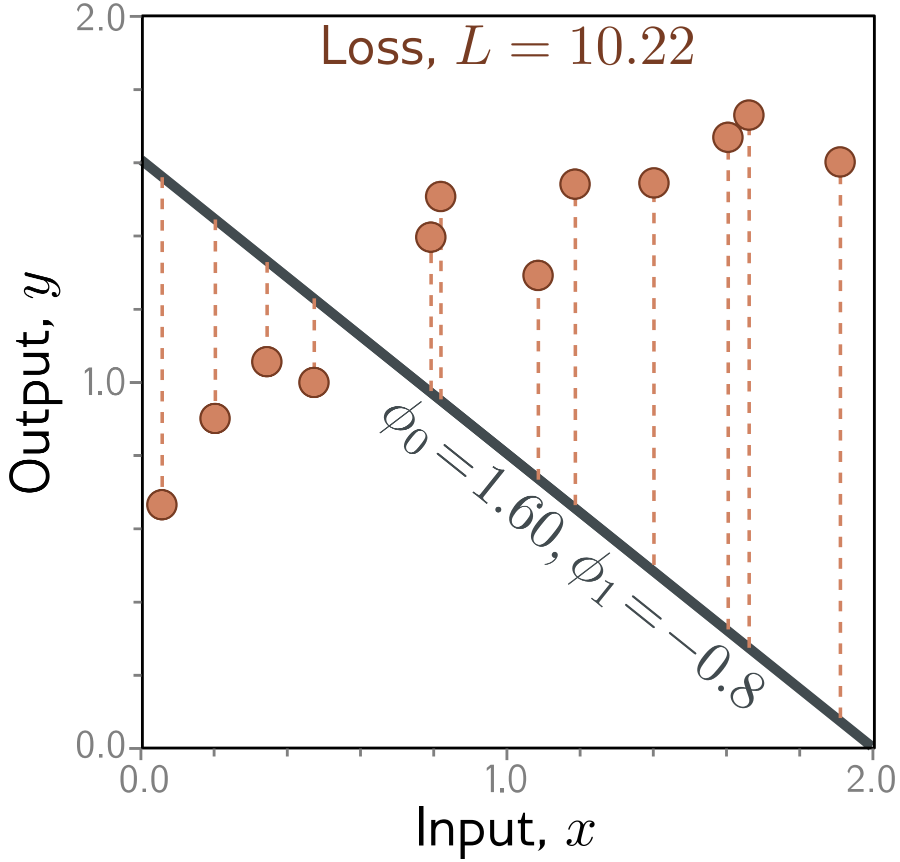
Example: 1D Linear regression loss function



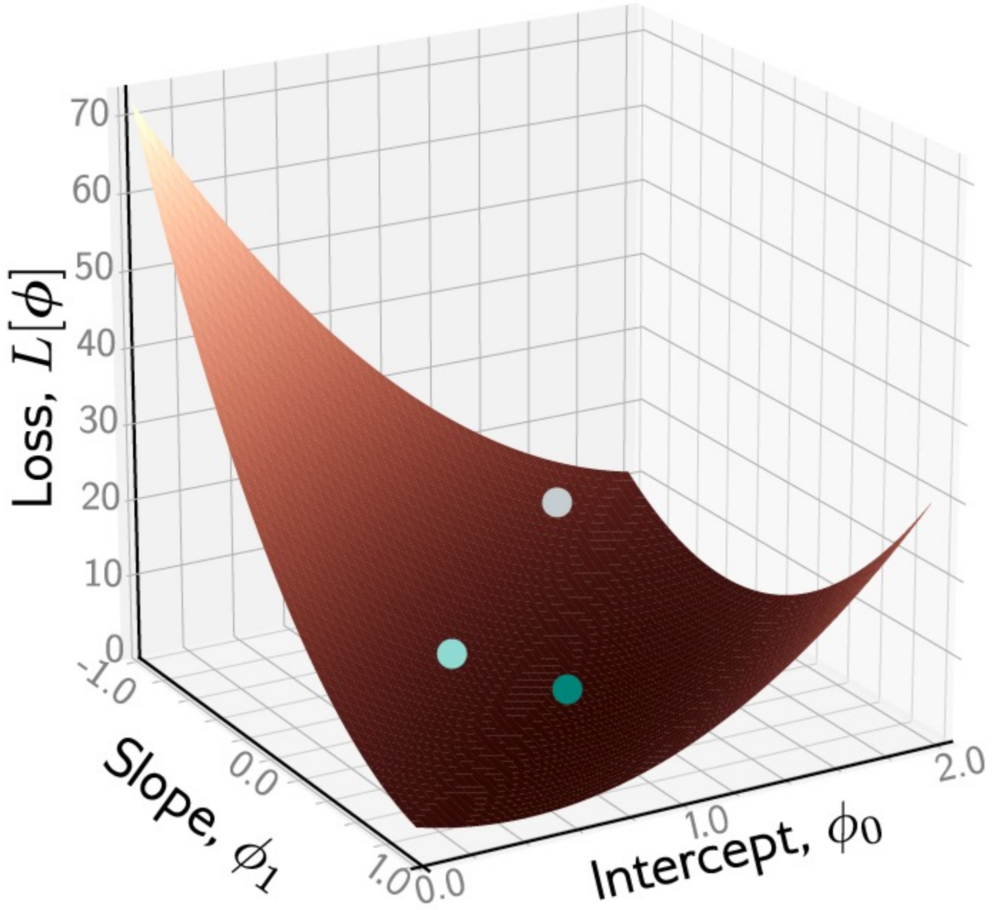


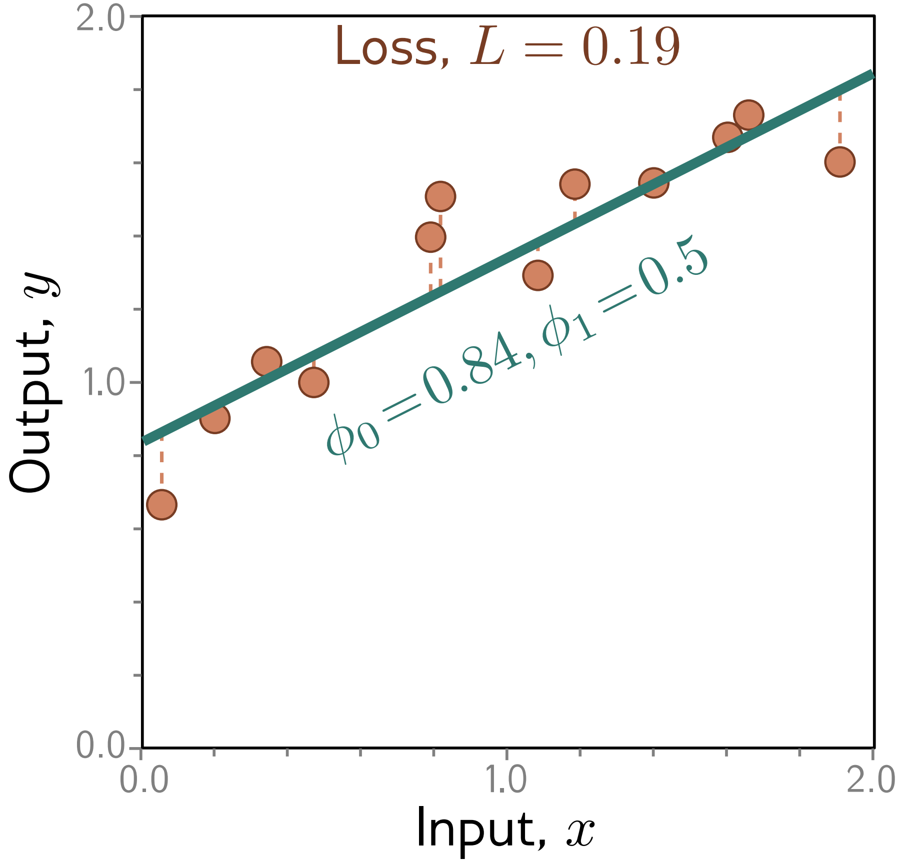
Example: 1D Linear regression loss function



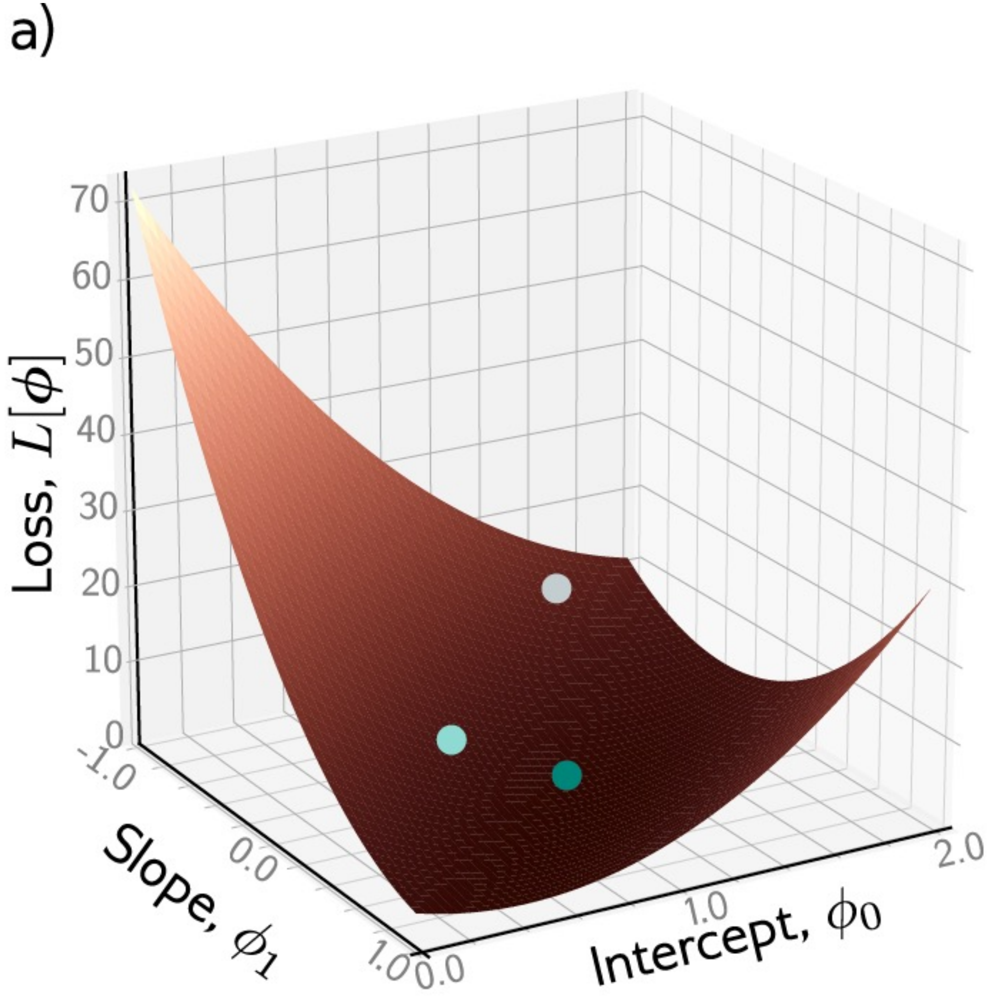


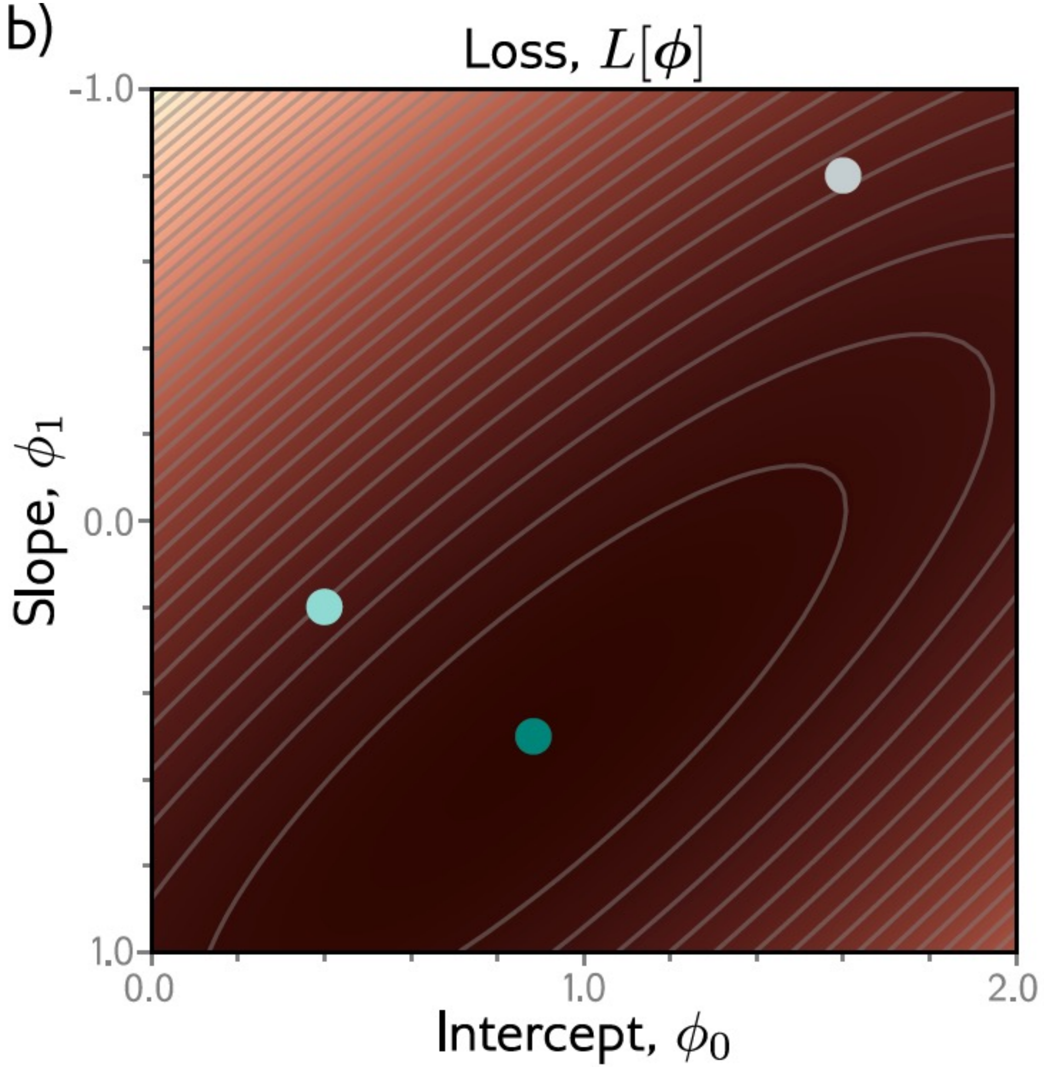
Example: 1D Linear regression loss function



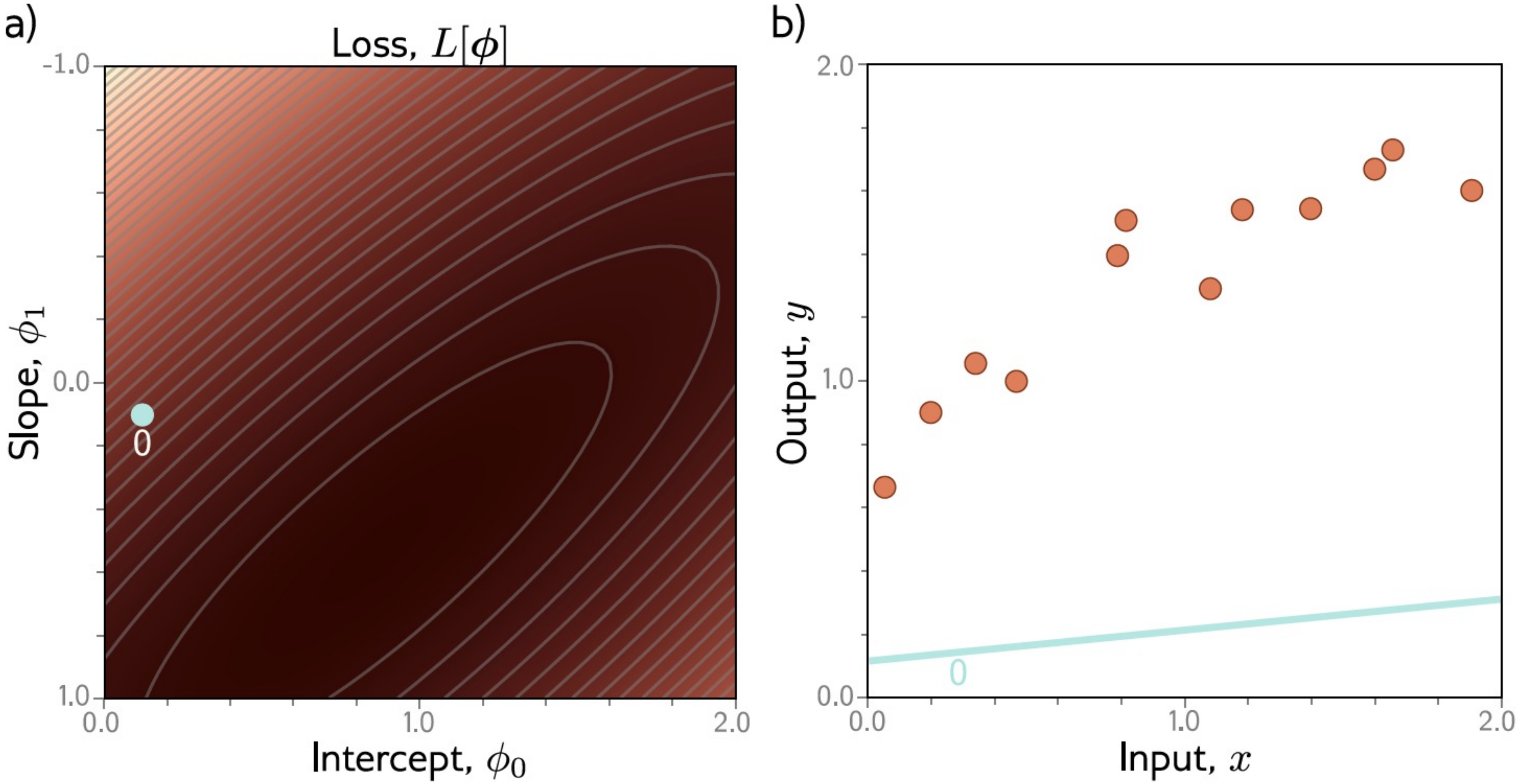


Example: 1D Linear regression loss function

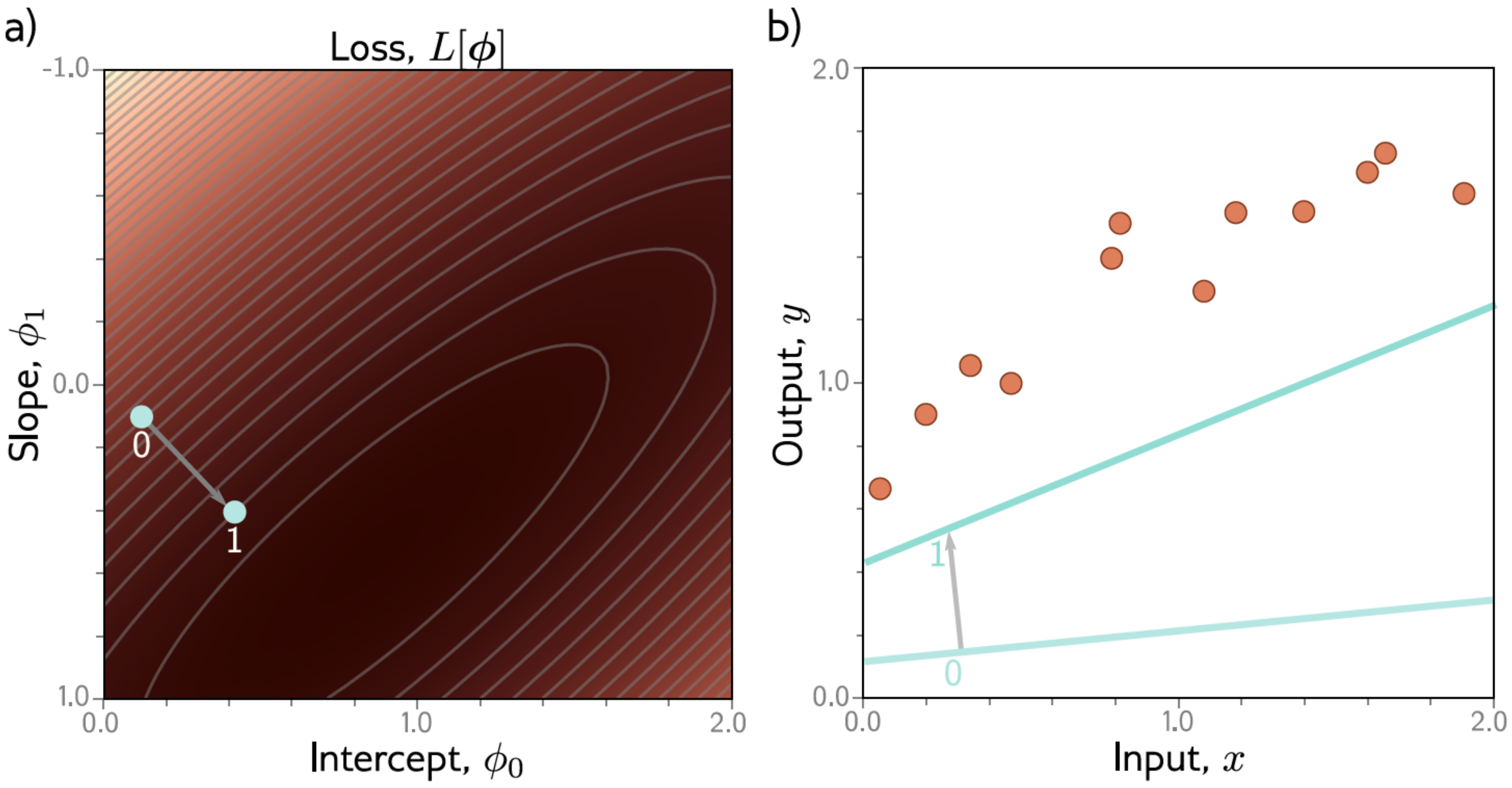




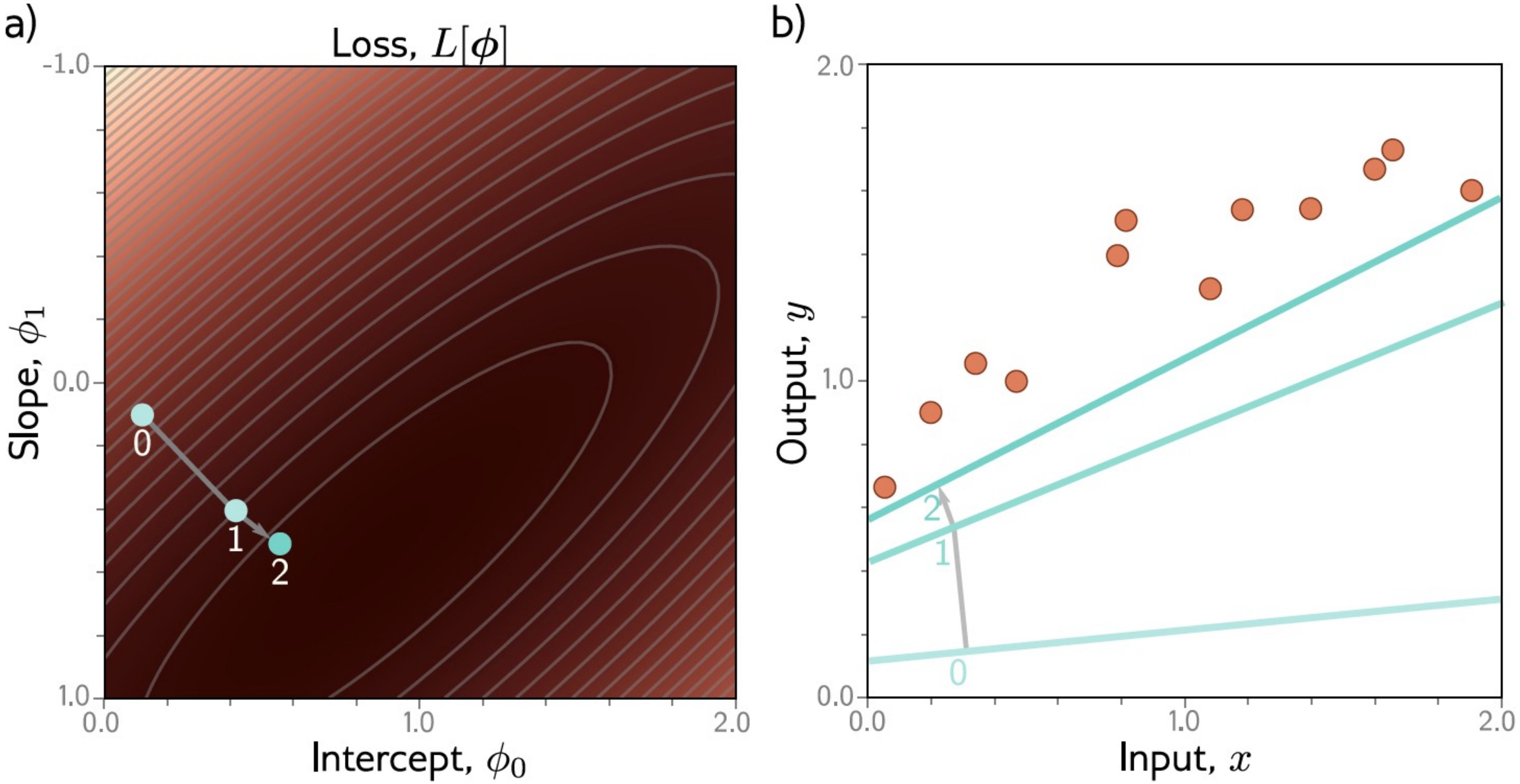
Example: 1D Linear regression training



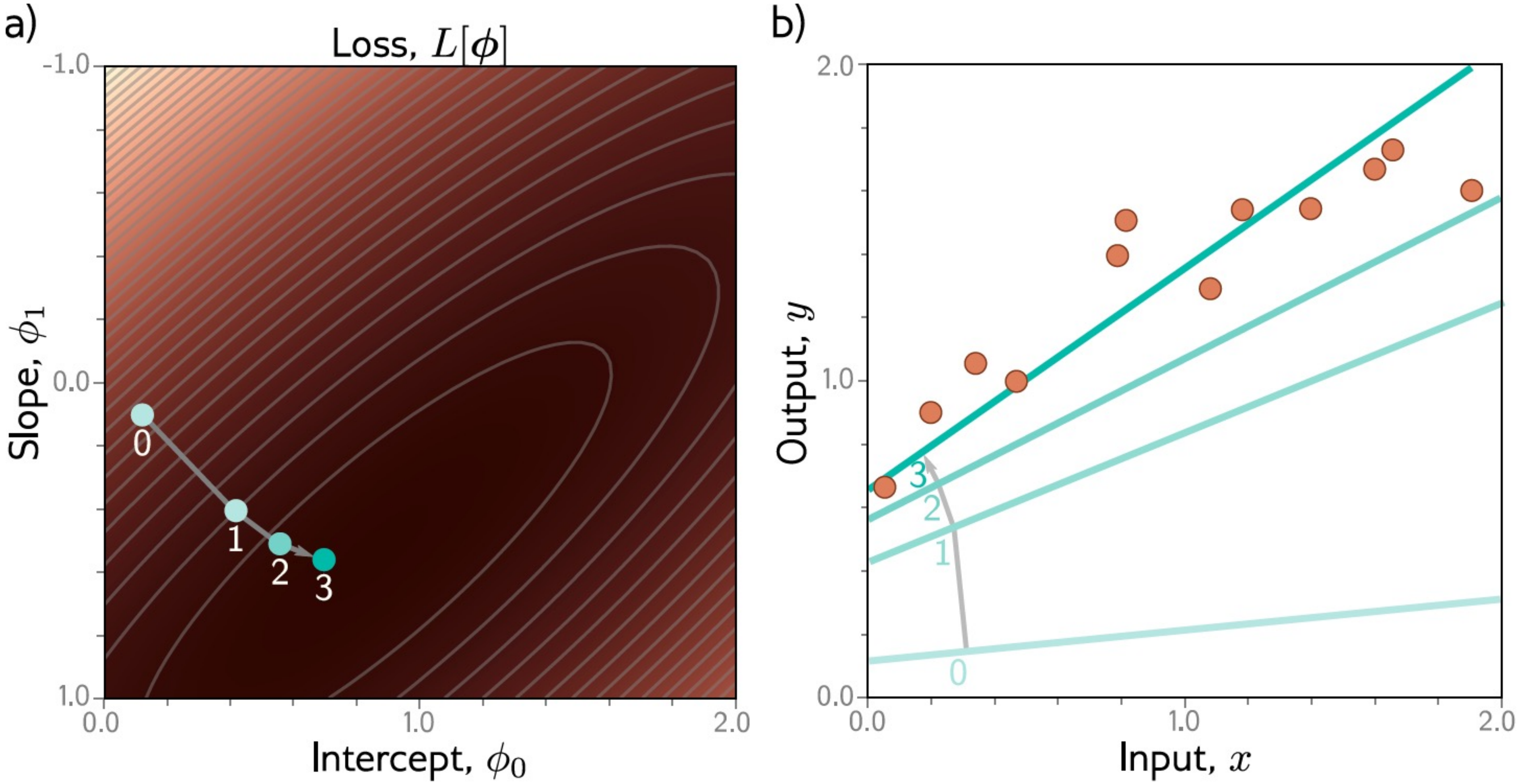
Example: 1D Linear regression training



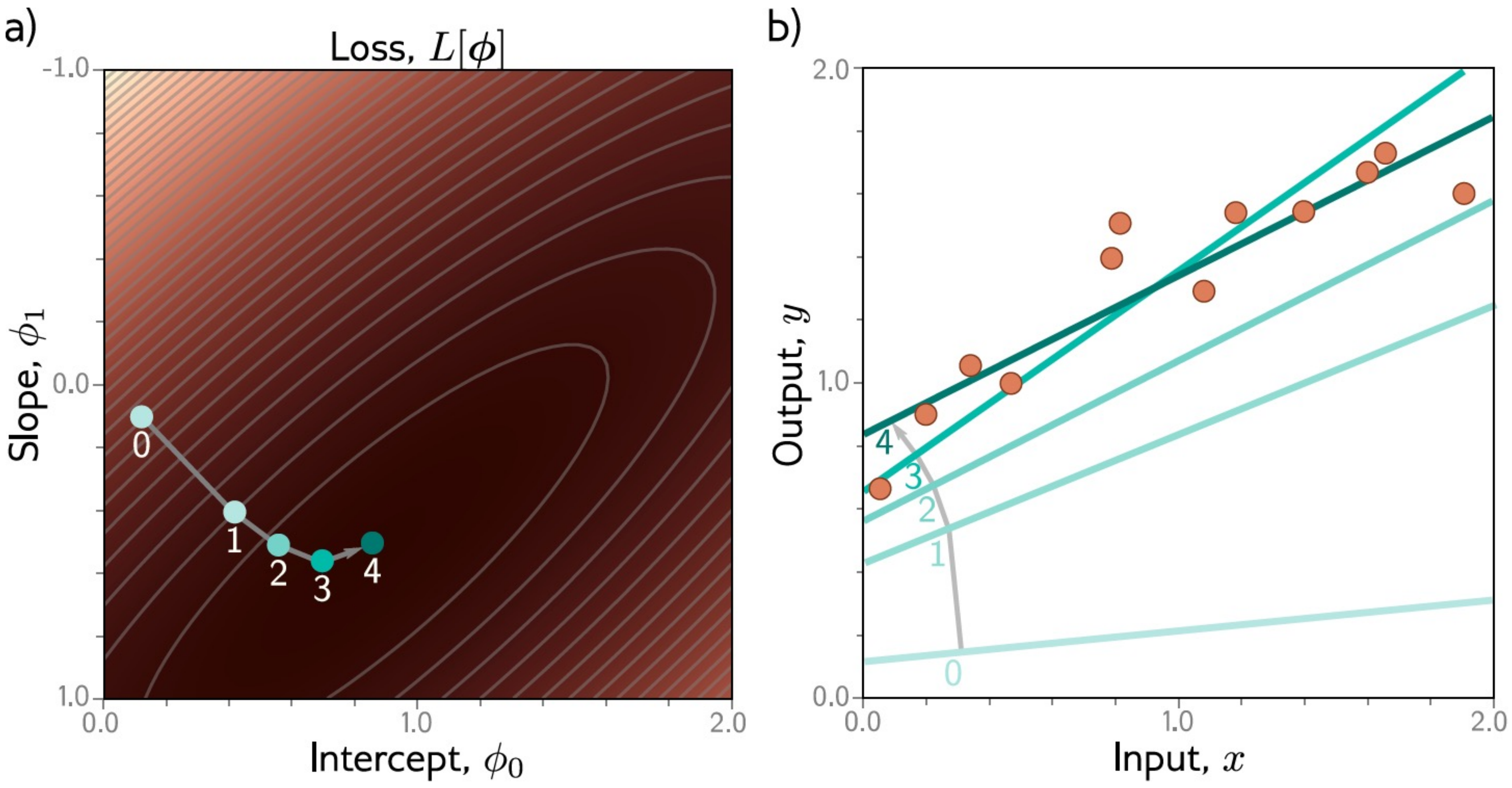
Example: 1D Linear regression training



Example: 1D Linear regression training



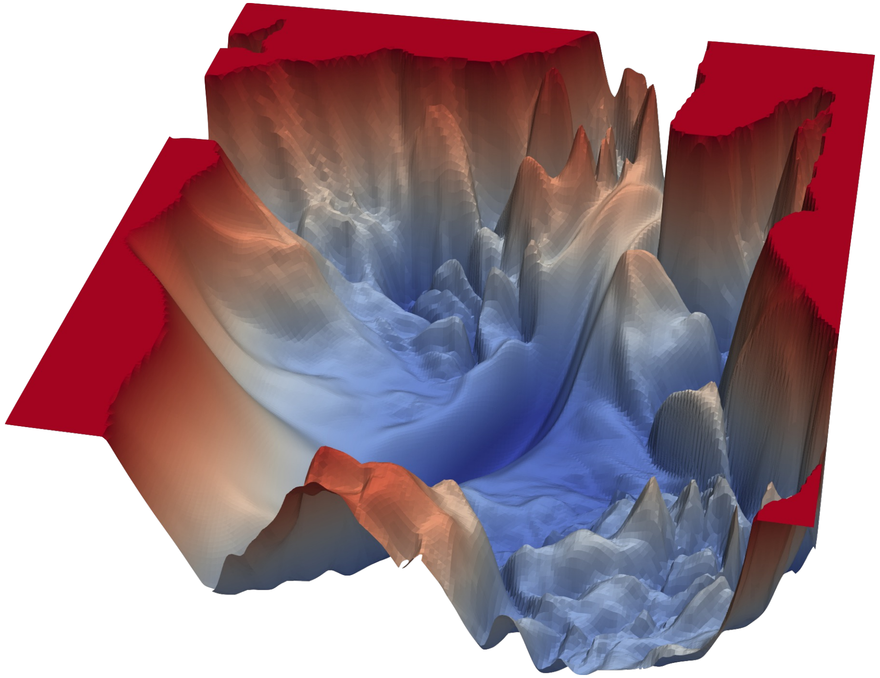
Example: 1D Linear regression training



This technique is known as gradient descent

Possible objections

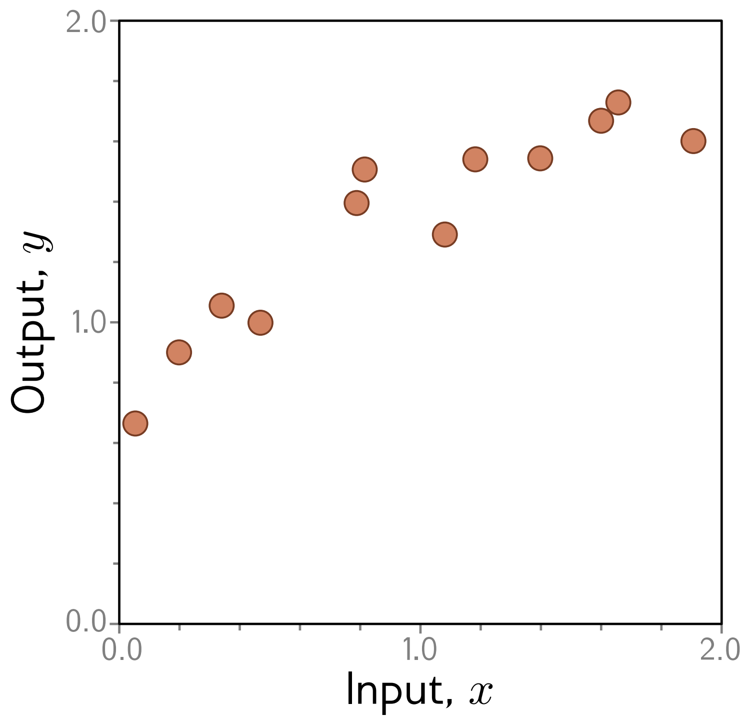
- But you can fit the line model in closed form!
  - Yes – but we won’t be able to do this for more complex models
- But we could exhaustively try every slope and intercept combo!
  - Yes – but we won’t be able to do this when there are a million parameters



Here's a visualization of the loss surface for the 56-layer neural network [VGG-56](http://arxiv.org/abs/1409.1556), from [Visualizing the Loss Landscape of Neural Networks](https://www.cs.umd.edu/~tomg/projects/landscapes/).

Example: 1D Linear regression testing

- Test with different set of paired input/output data (Test Set)
  - Measure performance
  - Degree to which *Loss* is same as training = generalization
- Might not generalize well because
  - Model too simple: underfitting
  - 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? Next lectures…

- Shallow neural networks (a more flexible model)
- Deep neural networks (even more flexible with fewer parameters)
- 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)

Course Project -- https://dl4ds.github.io/sp2024/project/

- Work in individually or in teams of 2-3
- Can be application, algorithmic, theoretical or combination thereof
- Some example ideas on the website, but propose new ones!
- Project proposal due Feb. 16
- Deliverables:
  - Code in GitHub repo
  - Report/paper
  - 3-4 minute video
- More info later, but feel free to brainstorm with me now

Possible Projects

- Class AI Tutor
- Teacher's AI Assistant
- CDS Curriculum AI Assistant
- CDS Building Recycling Advisor
- Media Bias Detection
- Herbaria Foundation Model
- Modern Implementation of Classic Models
- Develop a new dataset for a new class of problem and an initial model
- *…your ideas here…*

Look at Kaggle, Conferences, Workshops, Datasets….

Feedback?

Link