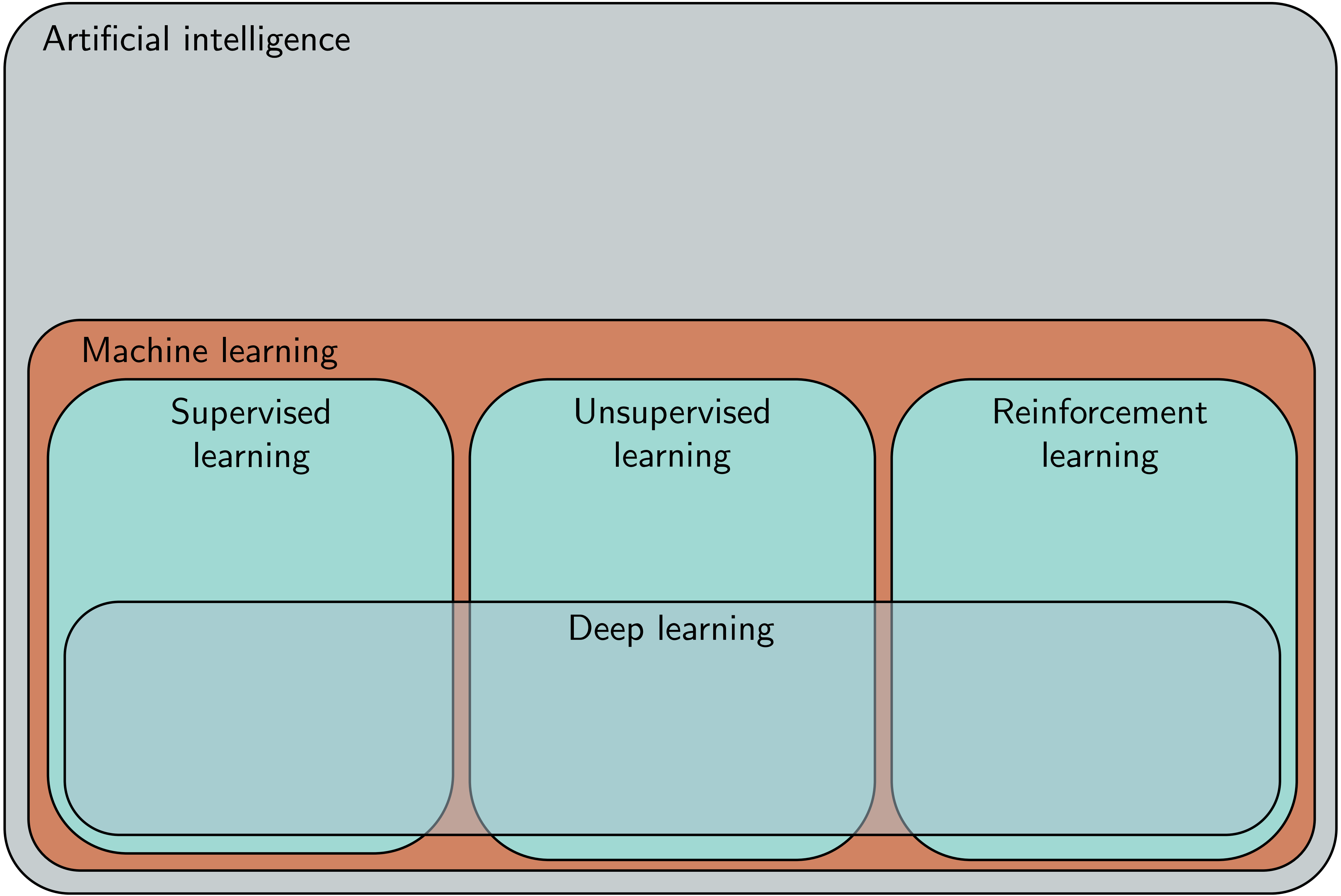
UNIVERSITY OF BATH

CM20315 - Machine Learning

Prof. Simon Prince

2. Supervised learning


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

Supervised learning

Bicycle

Apple

Aardvark

Supervised learning

Bicycle

Apple

Aardvark

Unsupervised learning

Supervised learning

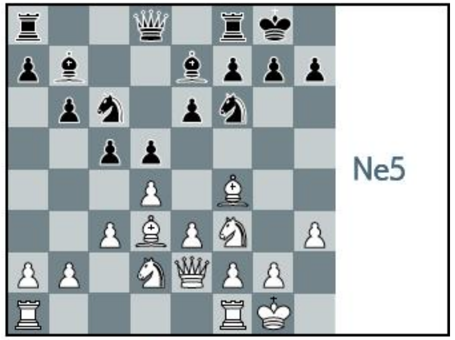
Bicycle

Apple

Aardvark

Unsupervised learning

Reinforcement learning



Reward $= 0$

Supervised learning

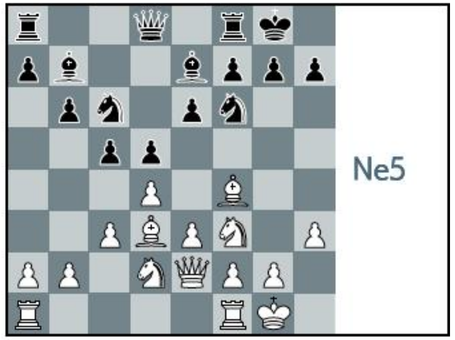
Bicycle

Apple

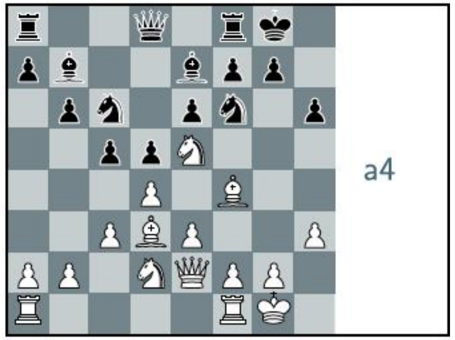
Aardvark

Unsupervised learning

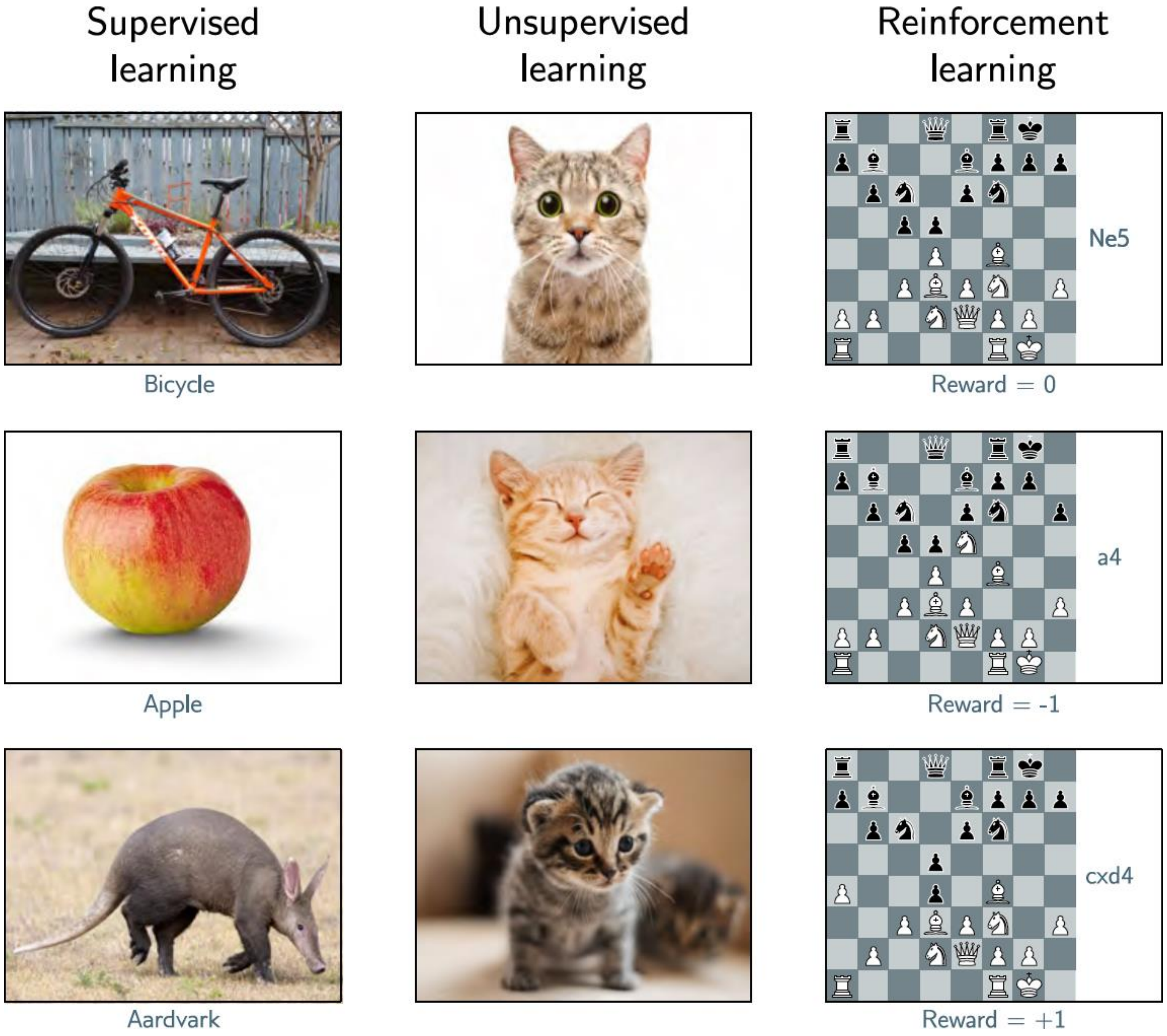
Reinforcement learning



Reward = 0



Reward = -1

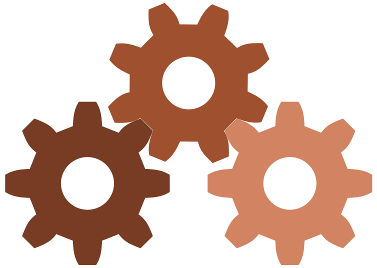

Supervised learning
Unsupervised learning
Reinforcement learning
Bicycle
Ne5
Reward = 0
Apple
a4
Reward = -1
Aardvark
cxd4
Reward = +1

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.

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



Supervised learning
model

$\rightarrow [340] \rightarrow$

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 ϕ 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.

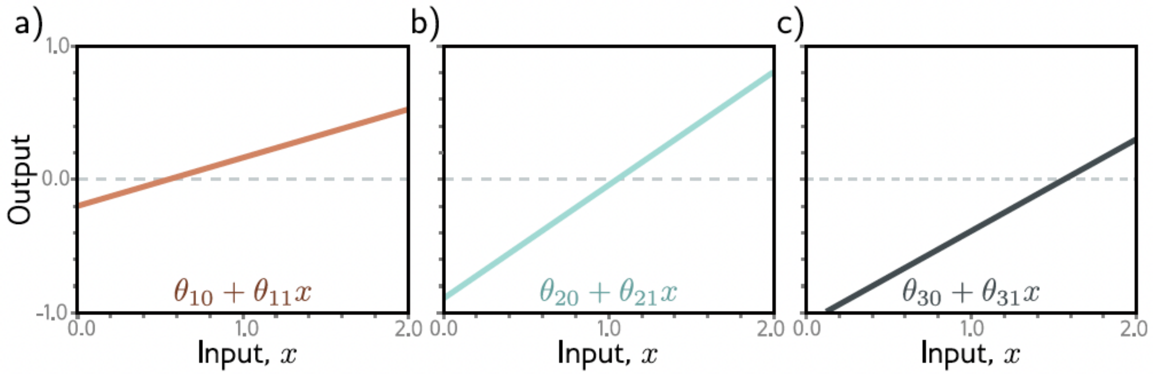


Figure 3.14 Processing in network with one input, three hidden units, and one output for problem 3.4. a–c) The input to each hidden unit is a linear function of the inputs. The first two are the same as in figure 3.3, but the last one differs.

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?

Appendix A

Notation

This appendix details the notation used in this book. This mostly adheres to standard conventions in computer science, but deep learning is applicable to many different areas, so it is explained in full. In addition, there are several notational conventions that are unique to this book, including notation for functions and the systematic distinction between parameters and variables.

Scalars, vectors, matrices, and tensors

Scalars are denoted by either small or capital letters a, A, α. Column vectors (i.e., 1D arrays of numbers) are denoted by small bold letters $\mathbf{a}, \boldsymbol{\phi}$, and row vectors as the transpose of column vectors $\mathbf{a}^T, \boldsymbol{\phi}^T$. Matrices and tensors (i.e., 2D and ND arrays of numbers, respectively) are both represented by bold capital letters $\mathbf{B}, \boldsymbol{\Phi}$.

Variables and parameters

Variables (usually the inputs and outputs of functions or intermediate calculations) are always denoted by Roman letters $a, \mathbf{b}, \mathbf{C}$. Parameters (which are internal to functions or probability distributions) are always denoted by Greek letters $\alpha, \boldsymbol{\beta}, \boldsymbol{\Gamma}$. Generic, unspecified parameters are denoted by $\boldsymbol{\phi}$. This distinction is retained throughout the book except for the policy in reinforcement learning, which is denoted by π according to the usual convention.

Sets

Sets are denoted by curly brackets, so $\{0, 1, 2\}$ denotes the numbers 0, 1, and 2. The notation $\{0, 1, 2, \ldots\}$ denotes the set of non-negative integers. Sometimes, we want to specify a set of variables and $\{\mathbf{x}_i\}_{i=1}^{I}$ denotes the I variables $\mathbf{x}_1, \ldots \mathbf{x}_I$. When it's not necessary to specify how many items are in the set, this is shortened to $\{\mathbf{x}_i\}$. The notation $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$ denotes the set of I pairs $\mathbf{x}_i, \mathbf{y}_i$. The convention for naming sets is to use calligraphic letters. Notably, $\mathcal{B}_t$ is used to denote the set of indices in a batch at iteration t during training. The number of elements in a set $\mathcal{S}$ is denoted by $|\mathcal{S}|$.

The set $\mathbb{R}$ denotes the set of real numbers. The set $\mathbb{R}^+$ denotes the set of non-negative real numbers. The notation $\mathbb{R}^D$ denotes the set of D-dimensional vectors containing real

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:

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

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:

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

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

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

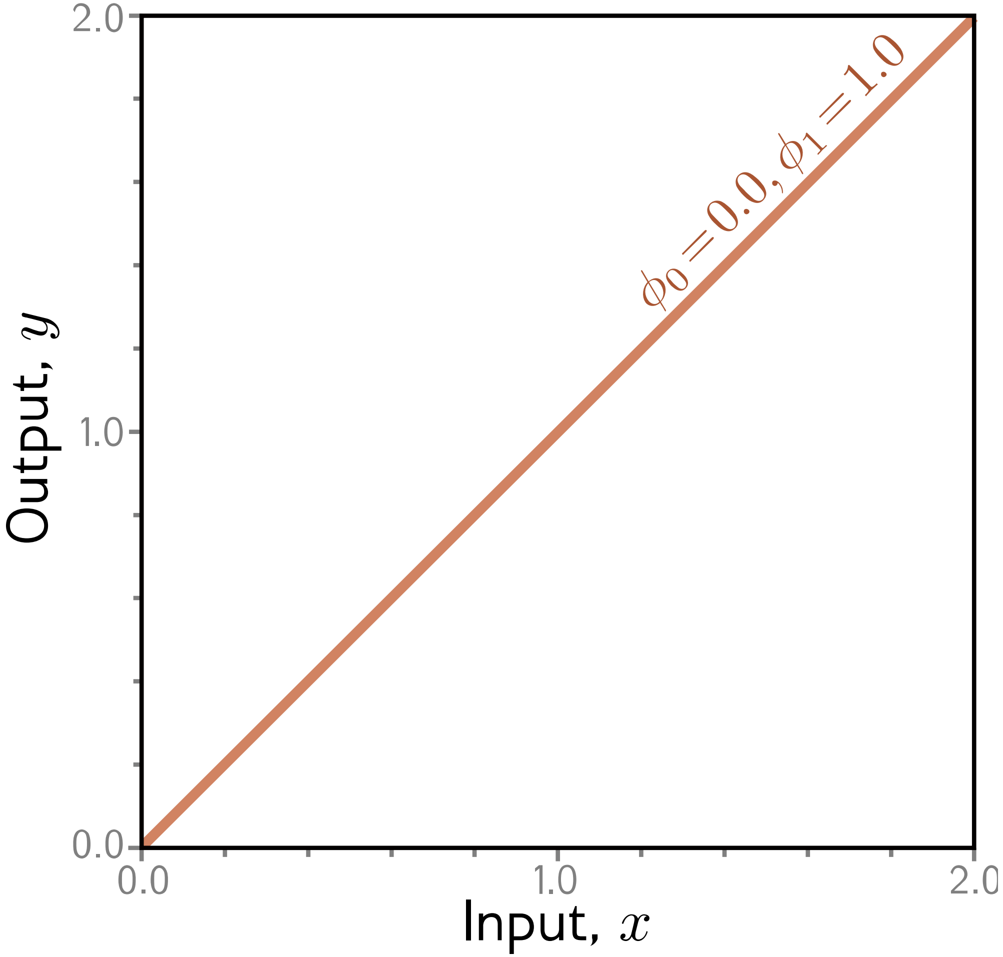
Example: 1D Linear regression model

- Model:

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

- Parameters

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



Example: 1D Linear regression model

- Model:

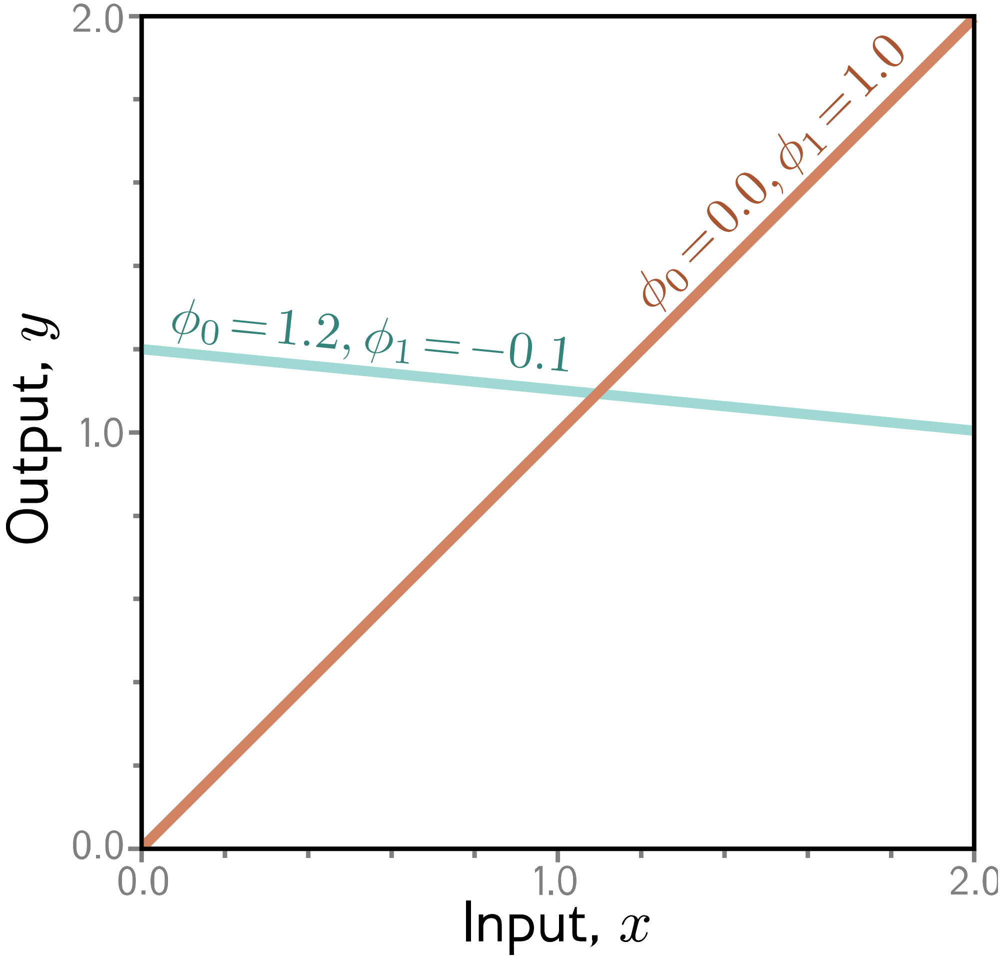
$$\begin{aligned} y &= \mathrm{f}[x, \boldsymbol{\phi}] \\ &= \phi_0 + \phi_1 x \end{aligned}$$

- Parameters

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

y-offset

slope



Example: 1D Linear regression model

- Model:

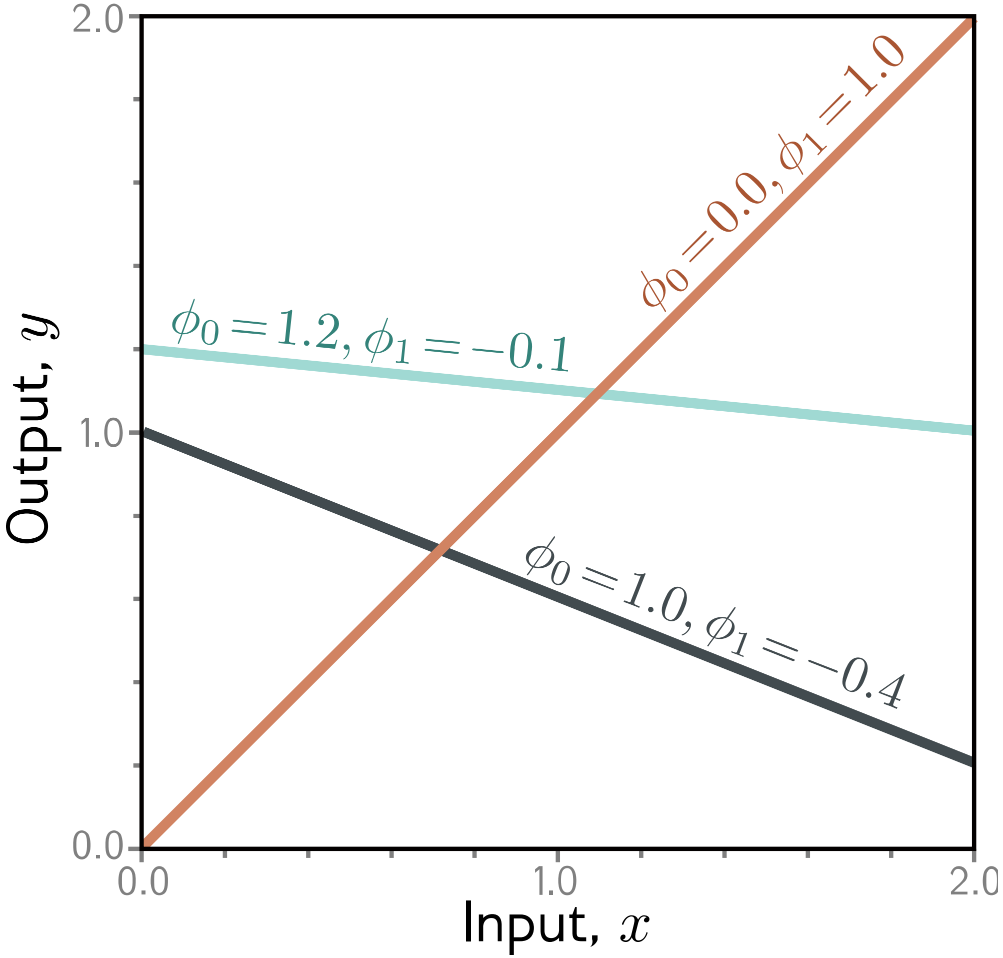
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

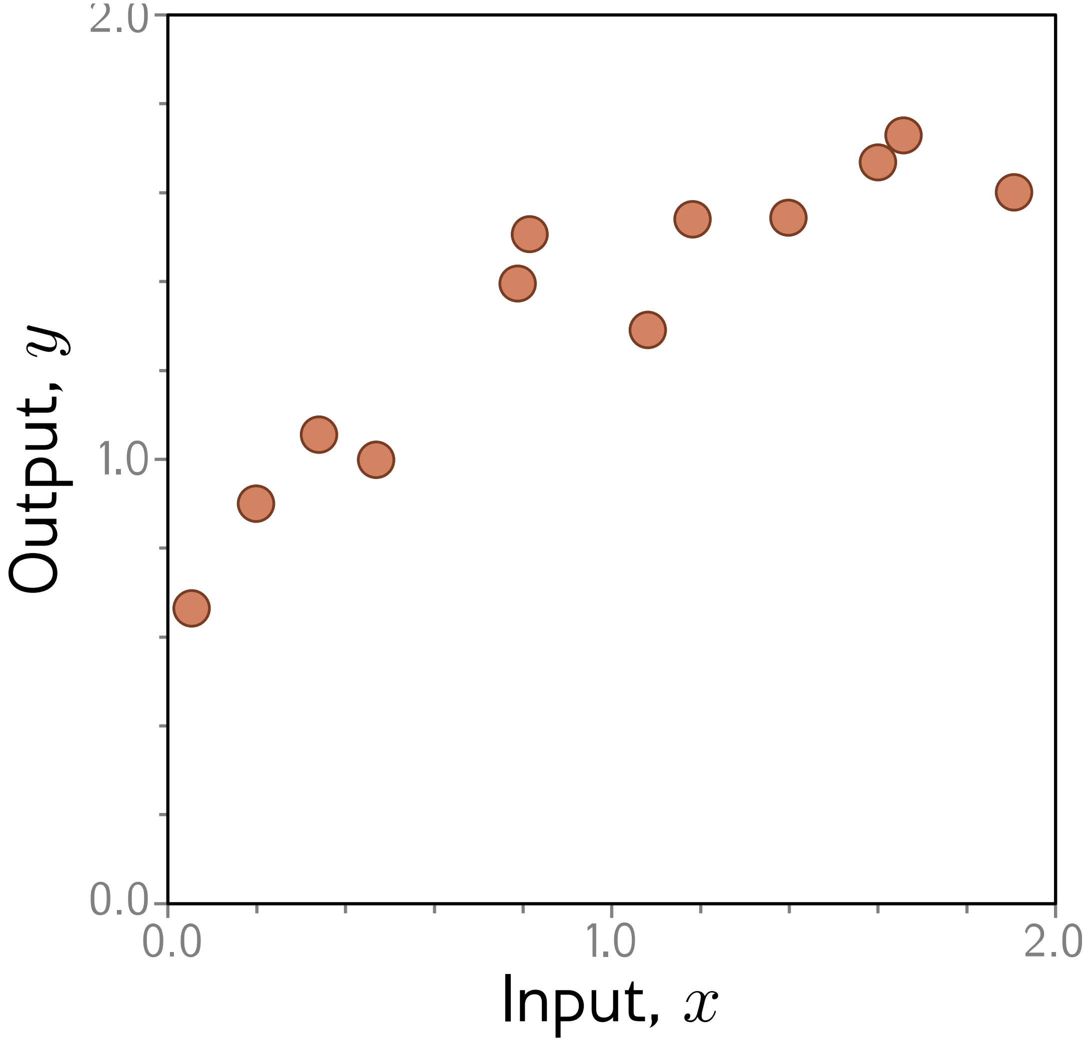
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

y-offset

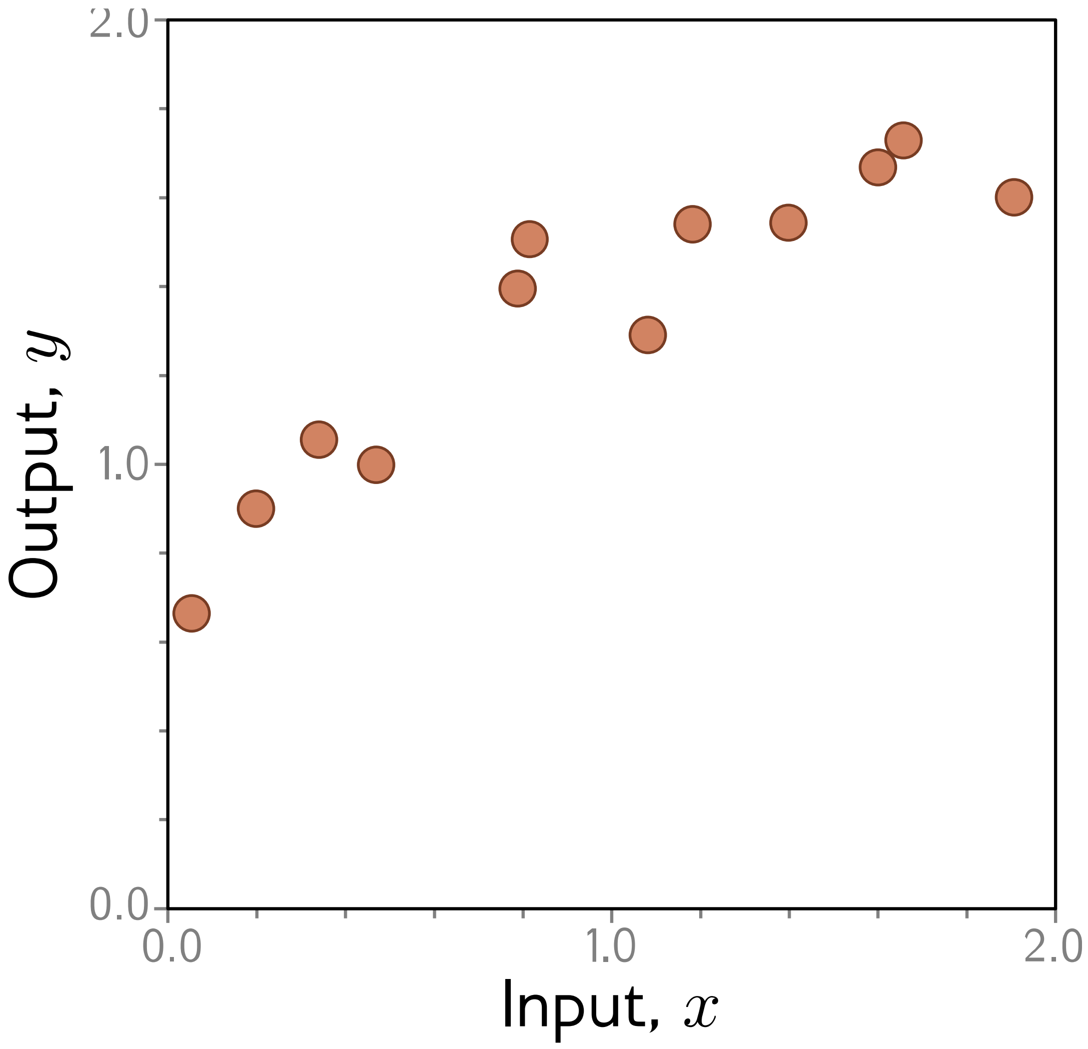
slope



Example: 1D Linear regression training data



Example: 1D Linear regression training data

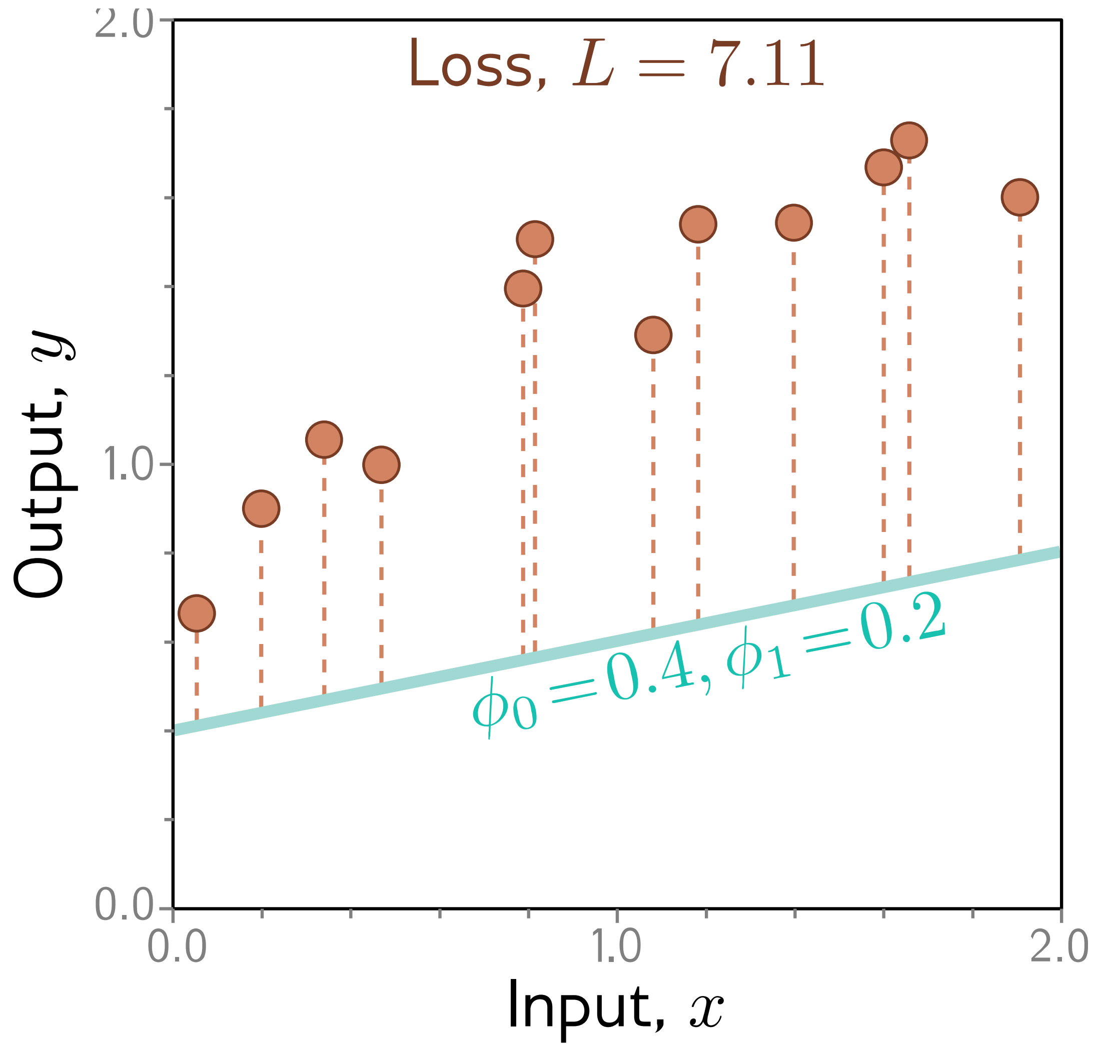


Loss function:

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

"Least squares loss function"

Example: 1D Linear regression loss function



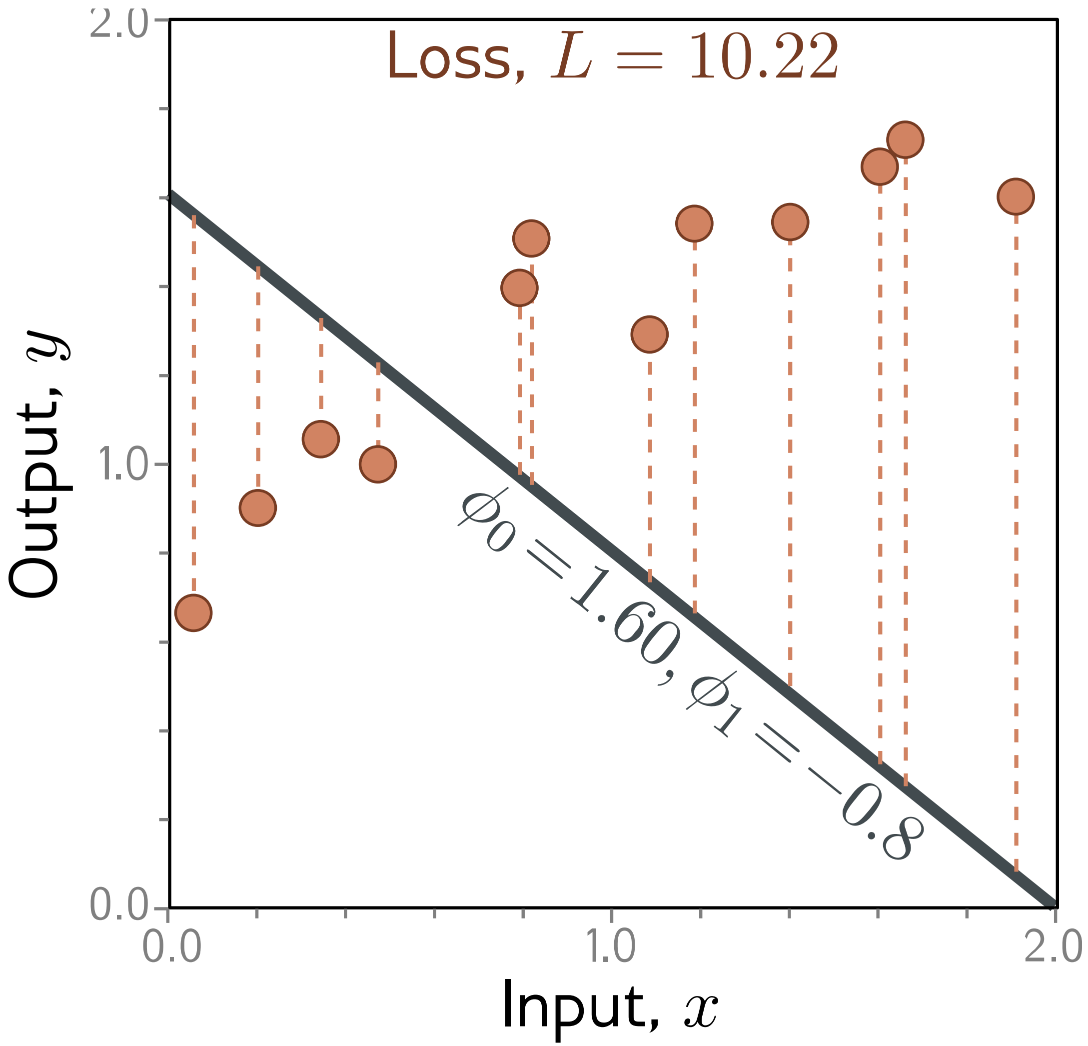
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



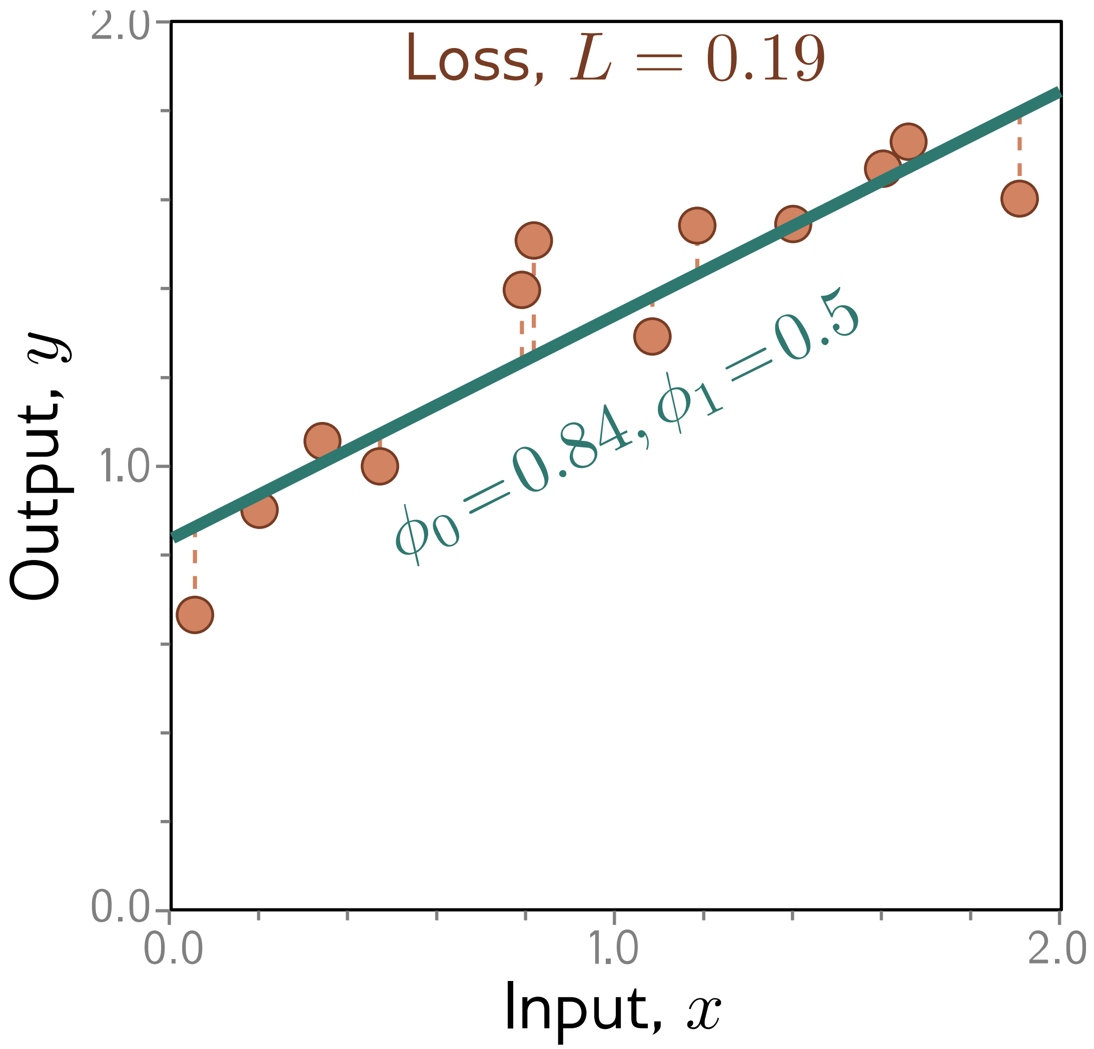
Loss function:

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

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

"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"

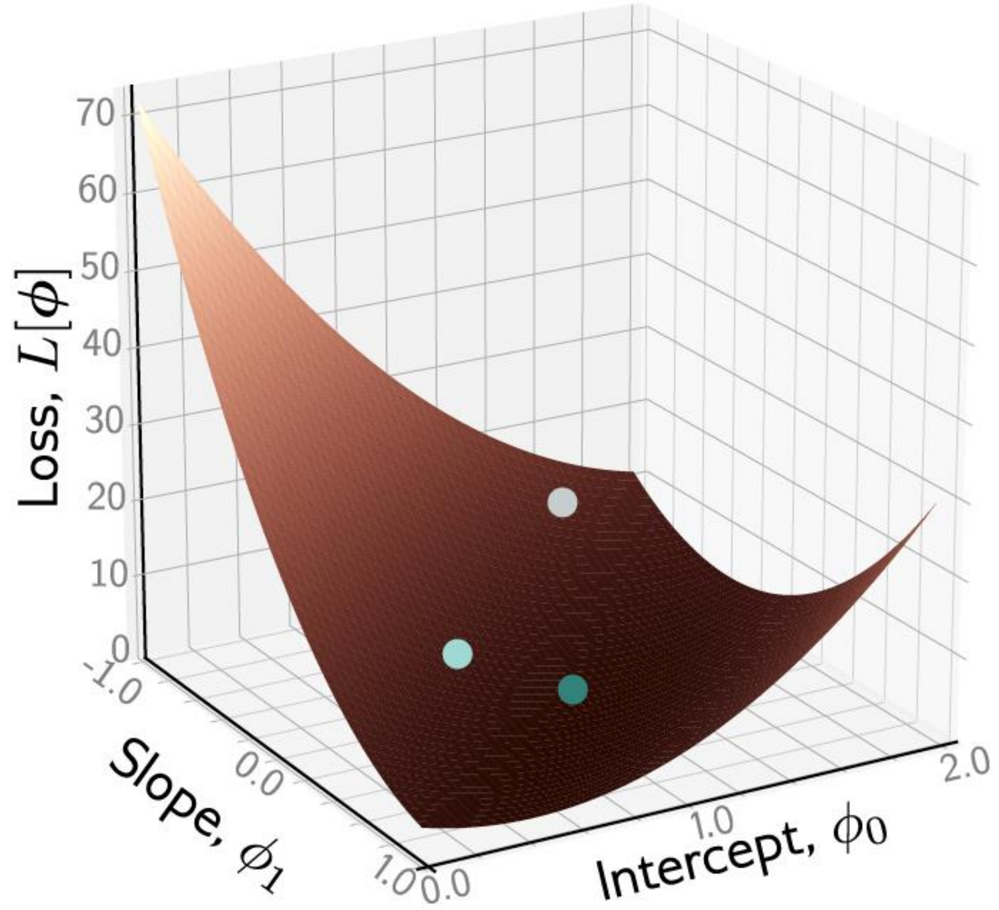
Example

- (0.1, 0.8), (0.9, 1.6), (1.4, 0.6) --> (x, y) from my dataset

Line equation mx+b , substitute by x to get y predicted

- 1- 0.9
- 2- 1.1
- 3- 0.6

Example: 1D Linear regression loss function



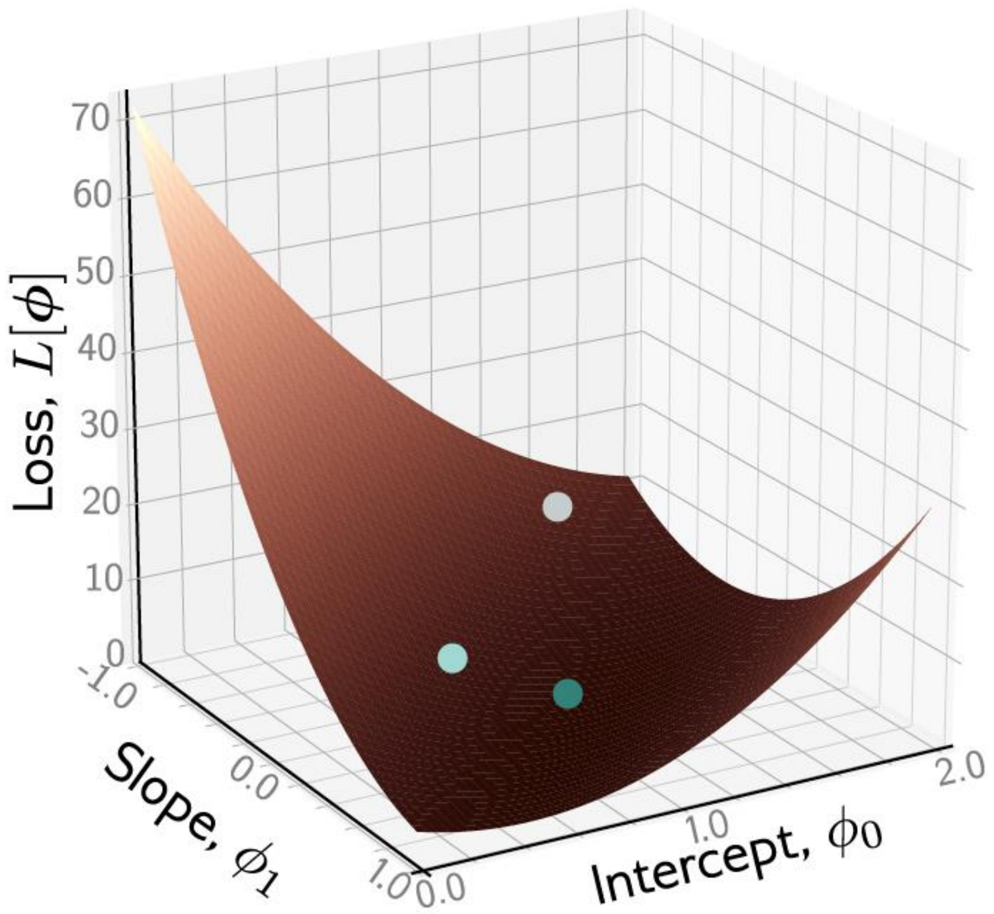
Loss function:

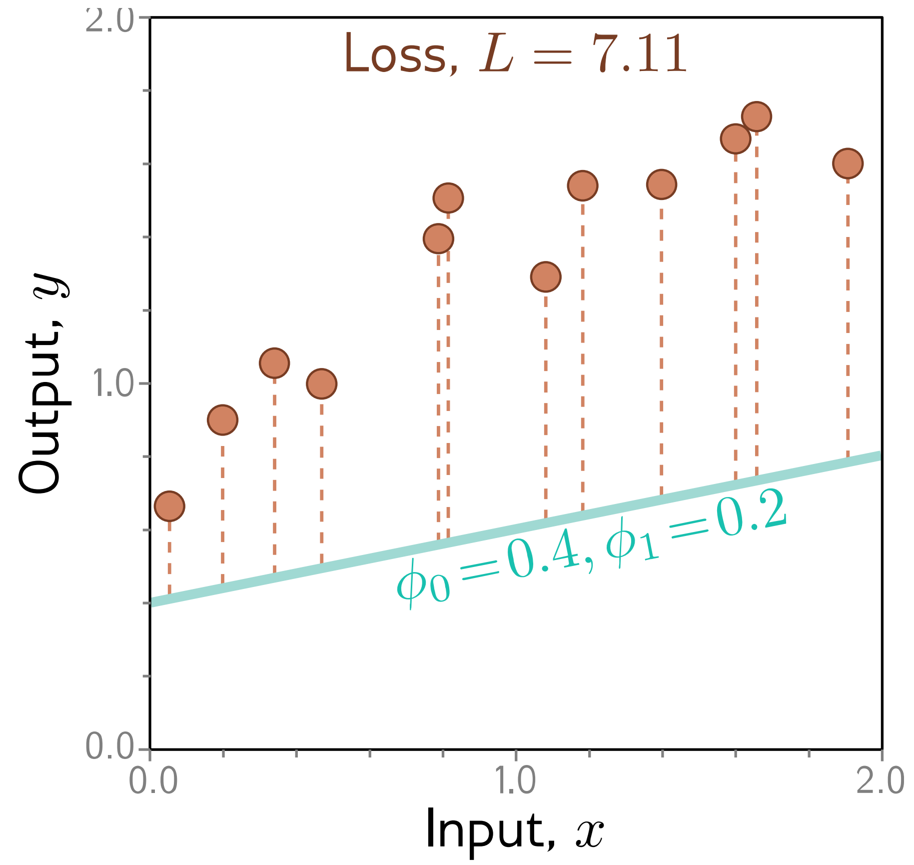
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

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

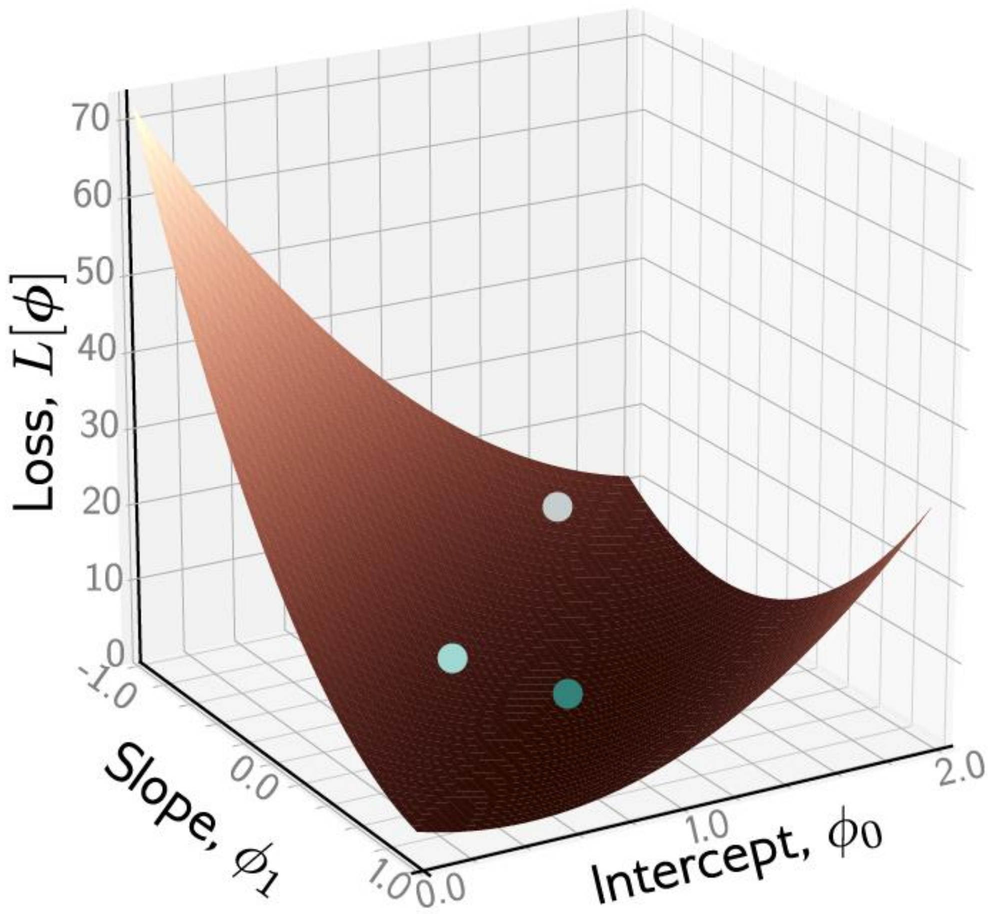
"Least squares loss function"

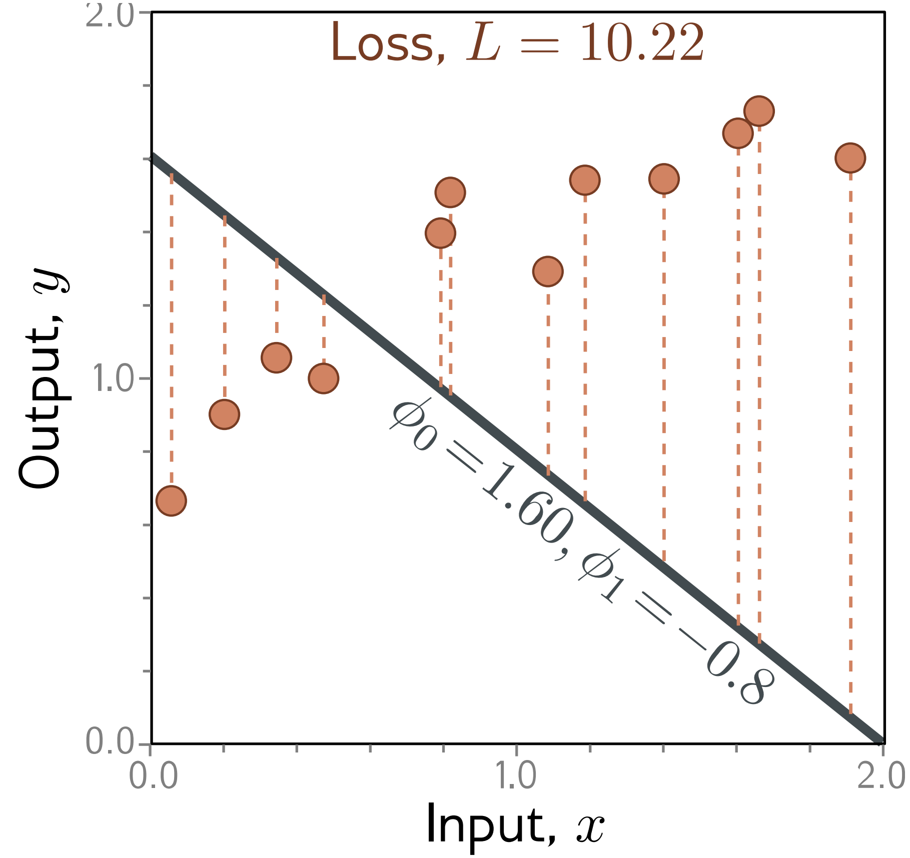
Example: 1D Linear regression loss function



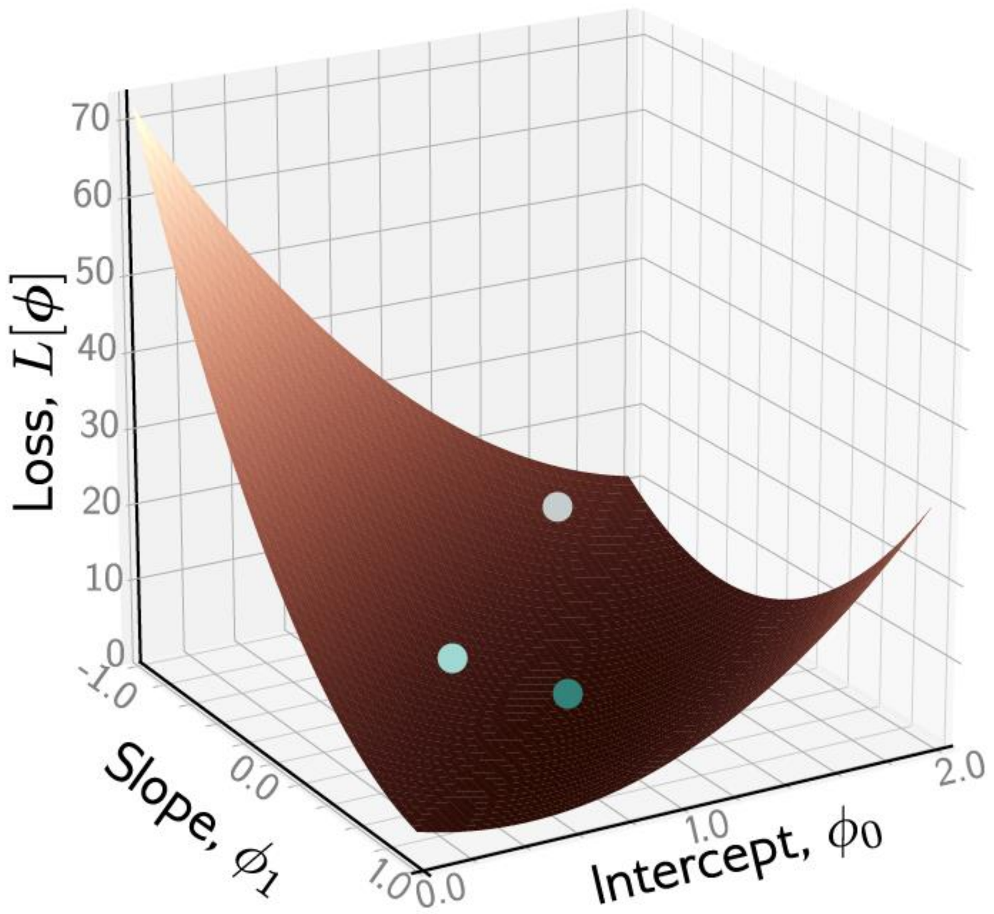


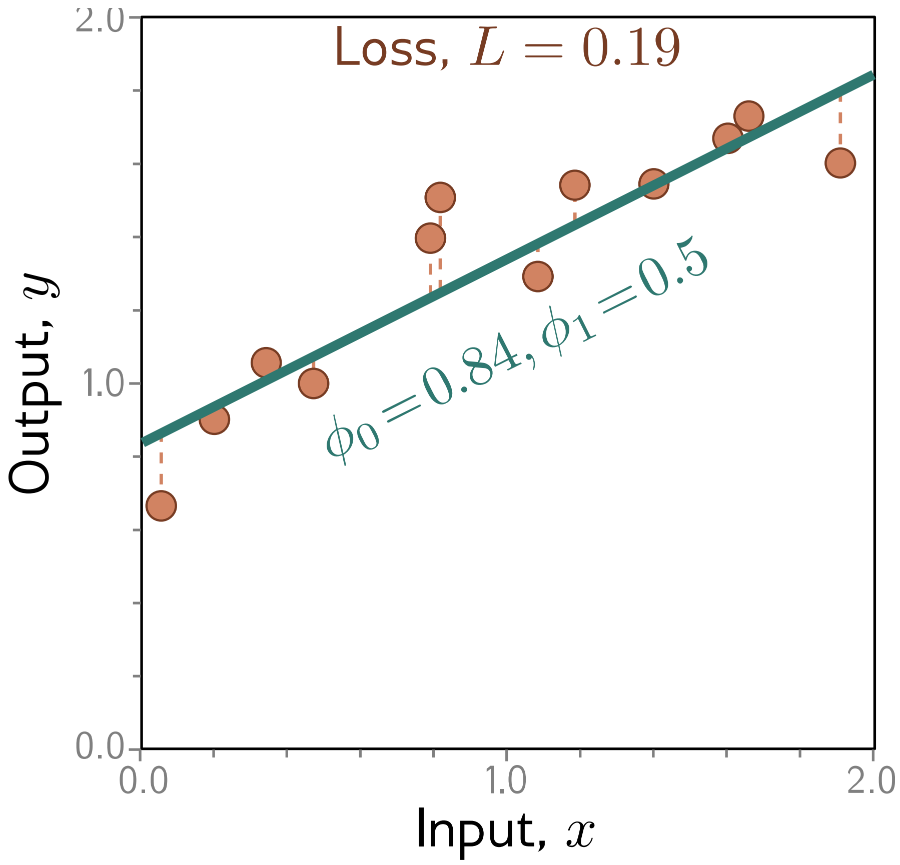
Example: 1D Linear regression loss function



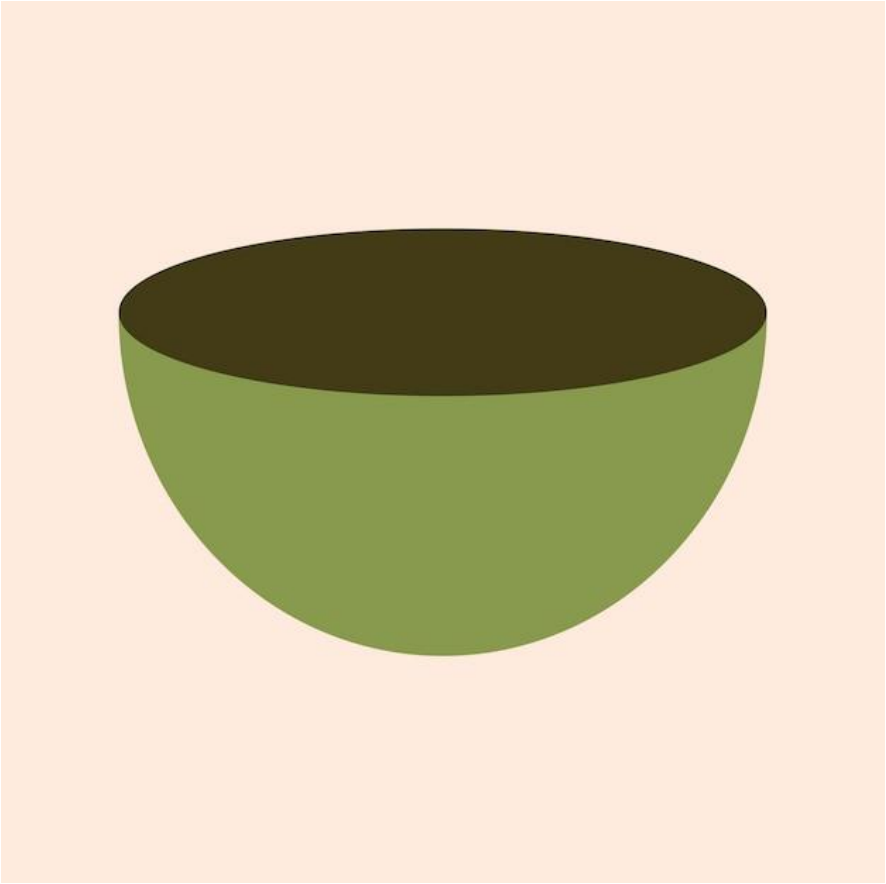


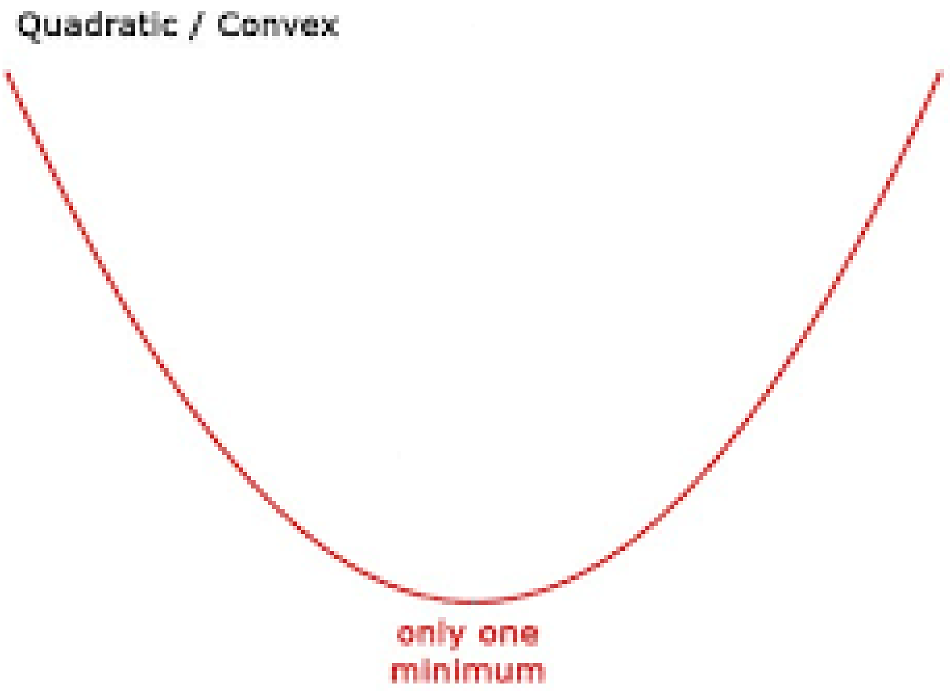
Example: 1D Linear regression loss function



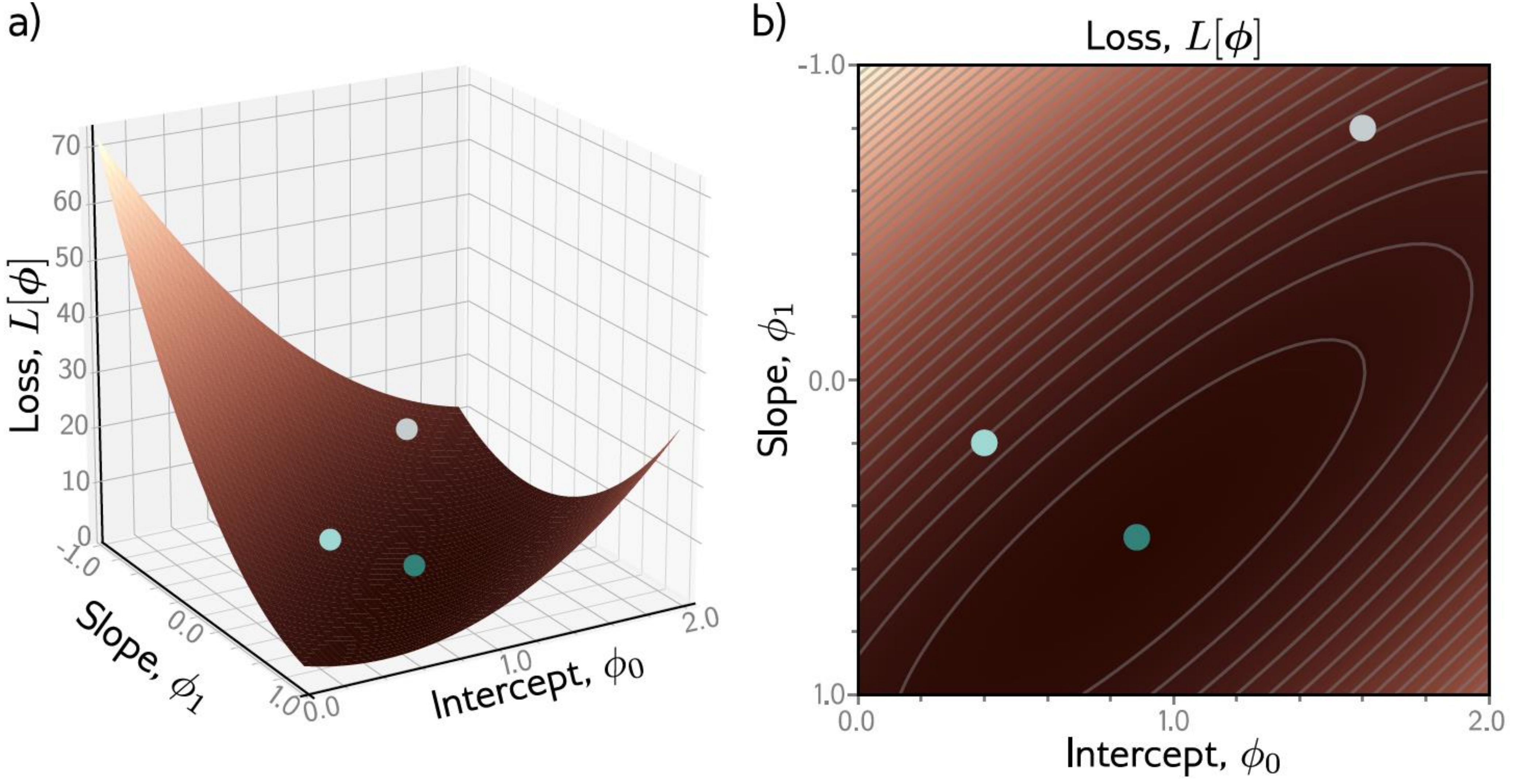


MSE – Mean Squared Error

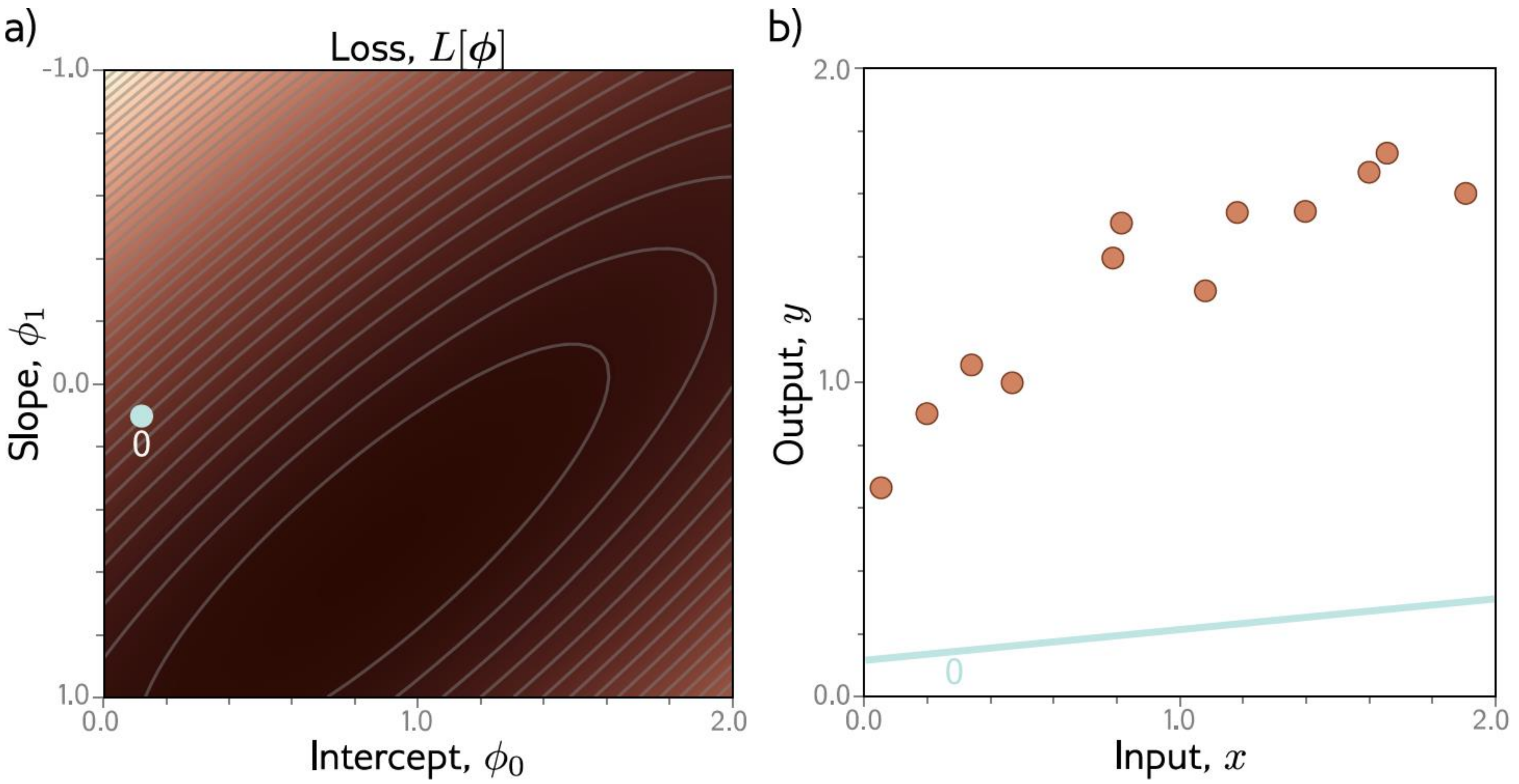




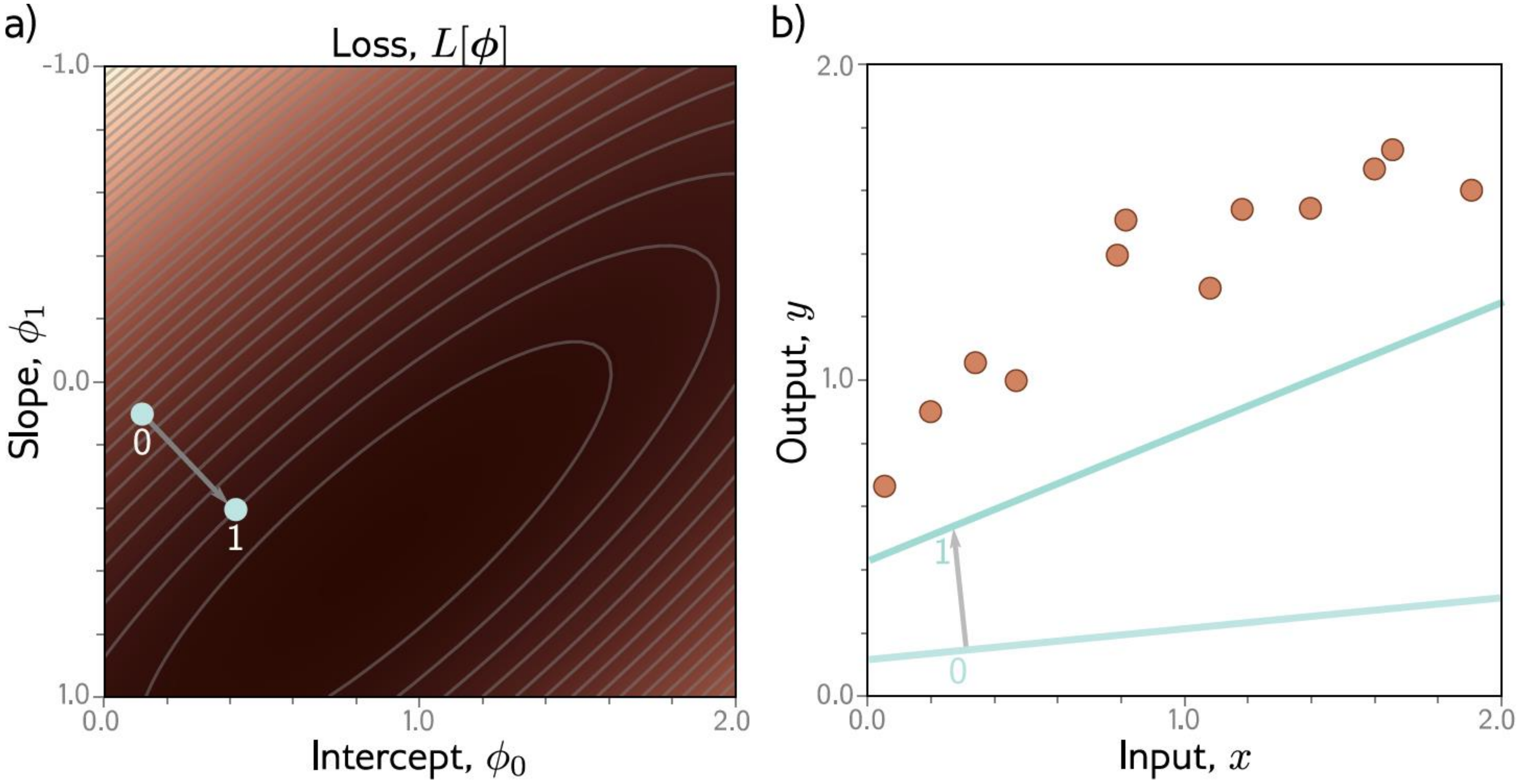
Example: 1D Linear regression loss function



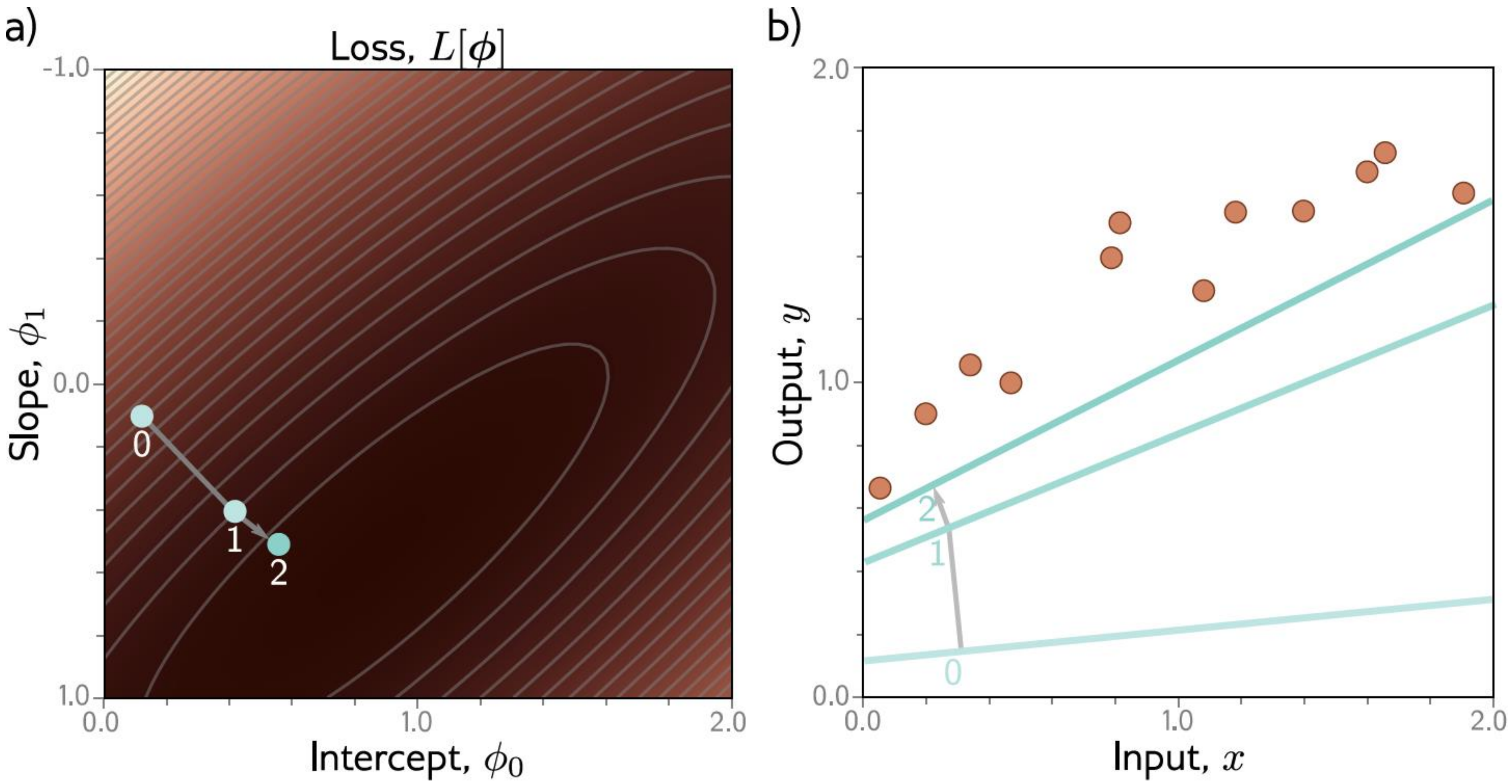
Example: 1D Linear regression training



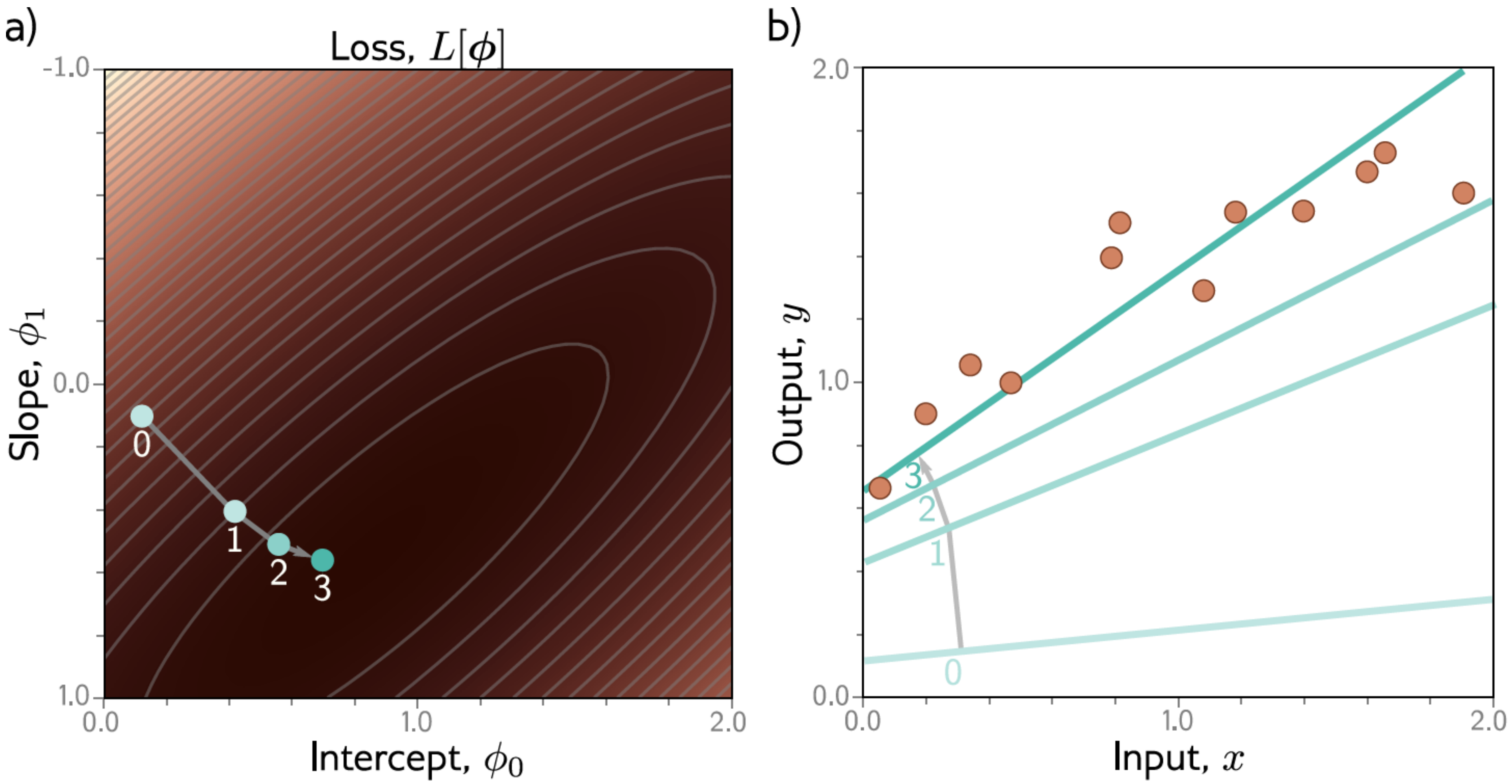
Example: 1D Linear regression training



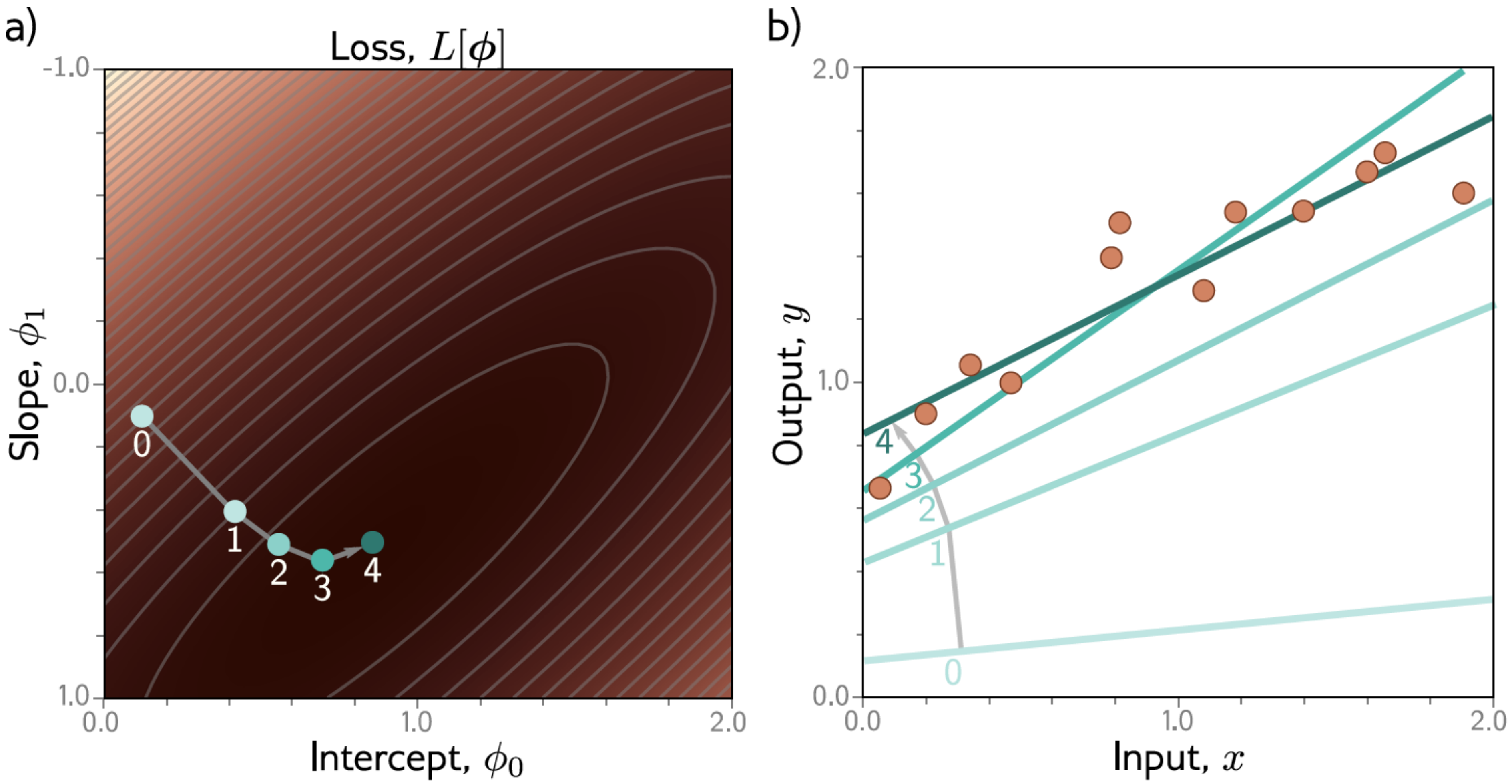
Example: 1D Linear regression training



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!

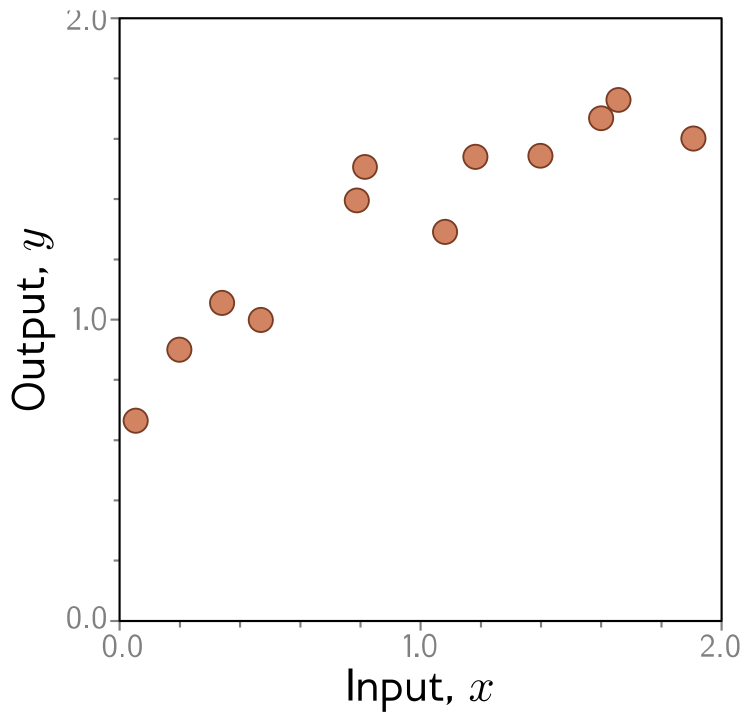
Yes – but we won't be able to do this for more complex models

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

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



Mentimeter!

- Go to this link: https://www.menti.com/alidaifuirtv

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)